

В.М. Пестриков

В.С. Дудкин

Г.А. Петров

ДИСКРЕТНАЯ МАТЕМАТИКА

УЧЕБНОЕ ПОСОБИЕ

**Санкт-Петербург
2013**

УДК 681.3 (075)

ББК 22.18. я 7

П 286

Дискретная математика: учеб. пособие /сост. В.М.Пестриков, В.С. Дудкин, Г.А. Петров. – СПб.: СПб ГТУРП, 2013.- 136 с

Учебное пособие посвящено изложению теории конечных автоматов и применению её методов для проектирования управляющих устройств и программ. Предназначено для самостоятельной работы студентов по дисциплине «Дискретная математика» направления «Прикладная математика и информатика» 010400.62 СПб ГТУРП очной формы обучения.

Рецензенты: канд. техн. наук., доцент кафедры «Прикладная математика и информатика» СПб ГТУРП Антонюк П.Е, канд. техн. наук., доцент кафедры «Информатика» СПб ГУСЭ Тихов С.В.

Рекомендовано к изданию Редакционно – издательским советом университета в качестве учебного пособия.

© Санкт-Петербургский
государственный технологический
университет растительных
полимеров, 2013

1. КОНЕЧНЫЕ АВТОМАТЫ

1.1. ВВЕДЕНИЕ

Все цифровые устройства можно разделить на два класса:

- логические схемы;
- схемы с памятью.

Логические и комбинационные схемы формируют значения выходных сигналов в зависимости от значений входных сигналов, подаваемых в данный момент независимо от значений входных сигналов, подаваемых на входы ранее. Для описания этих схем используются переключательные (булевы) функции.

В другом классе схем значение выходных сигналов зависят не только от значений входных сигналов, поступающих в данный момент, но еще и от значений сигналов, подаваемых на эти входы ранее, т.е. эти схемы учитывают предысторию входных сигналов. Эти схемы называются схемами с памятью или последовательностными схемами, так как они последовательность входных сигналов перерабатывают в последовательность выходных сигналов.

Для описания таких схем используется теория конечных автоматов.

1.2. ТЕОРИЯ КОНЕЧНЫХ АВТОМАТОВ

Основные определения:

- конечная совокупность различных символов называется *алфавитом*;
- каждый отдельный символ алфавита – *буквой*;
- последовательность букв, имеющая конечную длину – *словом*;
- число букв в слове – *длиной*;
- два алфавита называются *равнозначными*, если между буквами этих алфавитов можно установить взаимно-однозначное соответствие.

Пример конечного автомата:

Пусть: $P = \{p_1, p_2, p_3\}$ – входной алфавит, $W = \{w_1, w_2\}$ – выходной алфавит.

Будем рассматривать абстрактный конечный автомат в виде «черного ящика» с одним входом и одним выходом (рис. 2.1), т.е. анализируемое устройство описывается на уровне задания зависимости значений на его выходе от значений на его входе.

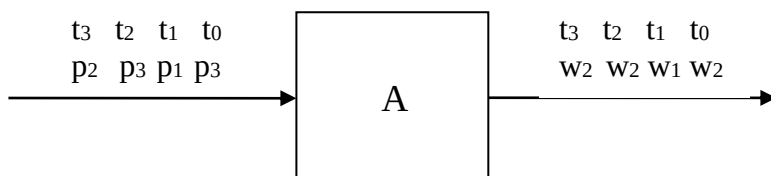


Рис. 1.1

Работа автомата рассматривается в дискретные моменты времени t_i называемые автоматными тактами (i меняется от 0 до n , где n – длина входного слова). В примере на рис.1.1 в начальный момент времени t_0 автомат входную букву p_3 перерабатывает в выходную букву w_2 . В результате работы автомата входное слово $p_3p_1p_3p_2$ перерабатывает в выходное слово $w_2w_1w_2w_2$.

В общем случае конечный автомат определяет устройство, которое реализует отображение множества слов входного алфавита во множество слов выходного алфавита.

Различают два класса абстрактных автоматов:

- автомат без выходного преобразователя;
- автомат с выходным преобразователем.

1.3. АВТОМАТ БЕЗ ВЫХОДНОГО ПРЕОБРАЗОВАТЕЛЯ

Автомат без выходного преобразователя – это следующая совокупность четырех объектов: $A = \langle P, S, s_0, \varphi \rangle$, где

$P = \{p_1, p_2, \dots, p_n\}$ – входной алфавит;

$S = \{s_0, s_1, s_2, \dots, s_k\}$ – множество состояний, в которых может находиться автомат;

s_0 – начальное состояние автомата, $s_0 \in S$;

φ – функция перехода автомата, которая представляет собой отображение декартова произведения множеств P и S на множество S , т.е. $P \times S \rightarrow S$.

Это значит, что каждой паре входной символ p_i и состояние s_j ставится в соответствие новое состояние s_k . Функция перехода записывается следующим образом:

$s_k = \varphi(p_i, s_j)$, где

p_i – входной символ;

s_j – текущее состояние автомата;

s_k – новое состояние автомата.

Абстрактный автомат называется конечным, если все его алфавиты конечны.

Автомат называется инициальным, если перед началом работы он должен быть установлен в начальное состояние s_0 .

Так как работа автомата рассматривается в дискретные моменты времени, то функцию перехода можно представить в виде:

$s_k(t+1) = \varphi(p_i(t), s_j(t))$, где

$p_i(t)$ и $s_j(t)$ – входной символ и состояние автомата в текущем автоматном такте (t);

$s_k(t+1)$ – новое состояние автомата в следующем автоматном такте ($t+1$).

Теория конечных автоматов не оперирует с такой величиной как время и символ t , в приведенной формуле функции перехода обозначает текущий автоматный такт, а $t+1$ обозначает следующий автоматный такт.

Обычно в теории автоматов обозначения автоматных тактов при записи функций не указываются.

Способы задания автомата без выходного преобразователя.

Пусть заданы: P – входной алфавит, S – множество состояний и s_0 – начальное состояние автомата.

Существует три способа задания функций перехода:

- перечислением;
- табличный;
- графический.

При перечислении приводятся все функции перехода:

$$s_1 = \varphi(s_0, p_1),$$

$$s_2 = \varphi(s_1, p_1),$$

.....

$$s_k = \varphi(s_{k-1}, p_n).$$

В табличном способе строится таблица переходов, столбцы которой помечаются входными символами, а строки – символами состояний из которых осуществляется переход. Внутри таблицы на пересечении i -той строки и j -того столбца указывается состояние, в которое переходит автомат из состояния s_i под воздействием входного символа p_j .

Таблица 1.1

$s_i \backslash p_j$	p_1	p_2		p_n
s_0	s_1	s_2		s_1
s_1	s_2	s_2		s_k
.....				
s_k				

В графическом способе каждому состоянию автомата ставится в соответствие вершина графа, которая помечается символом этого состояния. Если из состояния s_i существует переход в состояние s_j под воздействием входного символа p_k , то вершины s_i и s_j соединяются дугой, исходящей из s_i , а сама дуга помечается символом p_k , под воздействием которого осуществляется данный переход (рис.1.2). Направление дуги указывает направление перехода, поэтому граф автомата является ориентированным графом.

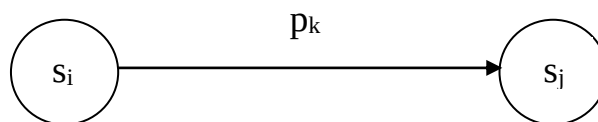


Рис. 1.2

Пример задания автомата без выходного преобразователя:

Пусть $P = \{p_1, p_2\}$, $S = \{s_0, s_1, s_2\}$

Функцию перехода зададим тремя различными рассмотренными способами:

Перечислением: $s_0 = \varphi(s_0, p_1)$, $s_1 = \varphi(s_0, p_2)$, $s_1 = \varphi(s_1, p_1)$, $s_2 = \varphi(s_1, p_2)$, $s_2 = \varphi(s_2, p_1)$, $s_0 = \varphi(s_2, p_2)$.

Таблицей переходов: (табл. 1.2).

Таблица 1.2

$s_i \backslash p_j$	p_1	p_2
s_0	s_0	s_1
s_1	s_1	s_2
s_2	s_2	s_0

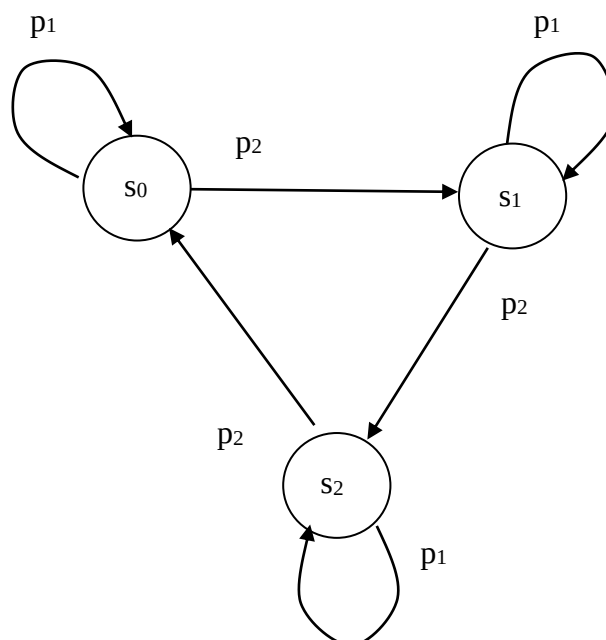


Рис. 1.3

Графом переходов: (рис. 1.3).

В табл.1.3 приводится моделирование работы автомата, приведенного в примере по переработке входного слова $p_1p_2p_2p_1p_2$. В процессе работы происходит следующая смена состояний автомата: $s_0 \rightarrow s_0 \rightarrow s_1 \rightarrow s_2 \rightarrow s_2 \rightarrow s_0$, т.е. работа автомата без выходного преобразователя состоит в получении последовательности состояний.

Таблица 1.3

Автоматный такт t_k	t_0	t_1	t_2	t_3	t_4	t_5
Состояние автомата s_i	s_0	s_0	s_1	s_2	s_2	s_0
Входной символ p_j	p_1	p_2	p_2	p_1	p_2	

1.4. АВТОМАТ С ВЫХОДНЫМ ПРЕОБРАЗОВАТЕЛЕМ

Автомат с выходным преобразователем это есть совокупность следующих шести объектов:

$A = \langle P, S, s_0, \varphi, W, \psi \rangle$, где

$P = \{p_1, p_2, \dots, p_n\}$ – входной алфавит;

$S = \{s_0, s_1, s_2, \dots, s_k\}$ – множество состояний;

s_0 – начальное состояние автомата, $s_0 \in S$;

φ – функция перехода автомата;

$W = \{w_1, w_2, \dots, w_m\}$ – выходной алфавит;

ψ – функция выхода.

Определение объектов P, S, s_0, φ совпадает с определением тех же объектов в автомате без выходного преобразователя.

Существует две математические модели автомата с выходным преобразователем: автомат Мура и автомат Мили. Определение функции выхода ψ различно в каждой из этих моделей. Оба автомата можно представить в виде композиции автомата без выходного преобразователя A^* и выходного преобразователя L .

Автомат Мура приведен на рис. 1.4, где $\tilde{P}, \tilde{S}, \tilde{W}$ – слова из соответствующих алфавитов.

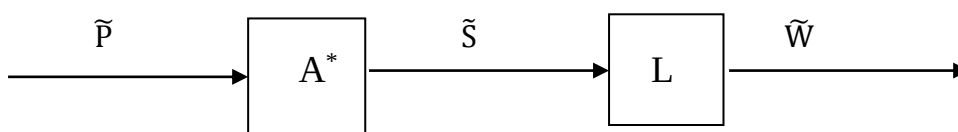


Рис.1.4

В автомате Мура функция выхода ψ осуществляет отображение множества S на множество W , т.е. $S \rightarrow W$. Это означает что каждому состоянию s_i ставится в соответствие выходной символ w_i . Функция перехода записывается следующим образом:

$w_i = \psi(s_i)$, где

s_i – текущее состояние автомата;

w_i – выходной символ.

Существенным в определении функции выхода автомата Мура является то, что новый выходной символ $w_i(t+1)$ в такте $(t+1)$ определяется следующим состоянием $s_i(t+1)$ в которое переходит автомат. Т.е.

$w_i(t+1) = \psi(s_i(t+1))$.

Автомат Мили приведен на рис. 1.5, где A^* – автомат без выходного преобразователя, L – преобразователь, $\tilde{P}, \tilde{S}, \tilde{W}$, – слова из соответствующих алфавитов.

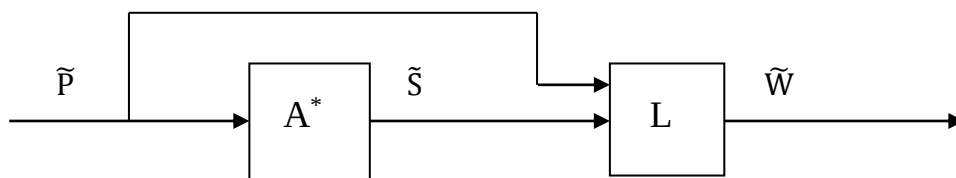


Рис.1.5

В автомате Мили функция выхода ψ осуществляет отображение декартова произведения множеств P и S на множество W , т.е. $P \times S \rightarrow W$. Это значит, что каждой паре входной символ p_i и состояние s_j ставится в соответствие выходной символ w_k . Функция выхода записывается следующим образом:

$w_k = \psi(p_i, s_j)$, где
 p_i – входной символ;
 s_j – текущее состояние автомата;
 w_k – выходной символ.

В функции выхода автомата Мили новый выходной символ $w_k(t+1)$ в такте $(t+1)$ определяется входным символом $p_i(t)$ и состоянием $s_i(t)$ в текущем такте (t) .

$$W_k(t+1) = \psi(p_i(t), s_j(t)).$$

Способы задания автомата Мура.

Пусть заданы: P – входной алфавит, S – множество состояний, s_0 – начальное состояние автомата и W – выходной алфавит.

Функции перехода и функции выхода могут быть представлены тремя способами:

- перечислением;
- табличный;
- графический.

При перечислении приводятся все функции перехода и выхода:

$$\begin{array}{ll} s_1 = \varphi(s_0, p_1), & w_1 = \psi(s_1), \\ s_2 = \varphi(s_1, p_1), & w_2 = \psi(s_2), \\ \dots\dots\dots & \dots\dots\dots \\ s_0 = \varphi(s_k, p_n). & w_0 = \psi(s_0). \end{array}$$

В табличном способе строится таблица переходов, которая слева дополняется столбцом, в котором для каждого состояния s_j указывается соответствующий ему выходной символ w_j (табл.1.4).

В графическом способе каждому состоянию автомата ставится в соответствие вершина графа, которая помечается символом этого состояния s_i и выходным символом w_i , соответствующим этому состоянию. Если из состояния s_i существует переход в состояние s_j под воздействием входного символа p_k , то вершины s_i и s_j соединяются дугой исходящей из s_i , а сама дуга помечается символом p_k под воздействием, которого осуществляется данный переход (рис.1.6).

Таблица 1.4

w_j	$\begin{matrix} p_i \\ s_j \end{matrix}$	p_1	p_2		p_n
w_0	s_0	s_1	s_2		s_1
w_1	s_1	s_2	s_2		s_k
.....					
w_k	s_k				s_0

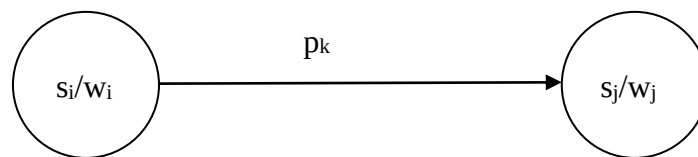


Рис. 1.6

Пример автомата Мура.

Пусть $P = \{p_1, p_2\}$, $W = \{w_0, w_1\}$, $S = \{s_0, s_1, s_2, s_3\}$.

Функции перехода :

$$s_0 = \varphi(s_0, p_1),$$

$$s_1 = \varphi(s_0, p_2),$$

$$s_2 = \varphi(s_1, p_1),$$

$$s_2 = \varphi(s_2, p_1),$$

$$s_3 = \varphi(s_1, p_2),$$

$$s_3 = \varphi(s_2, p_2),$$

$$s_3 = \varphi(s_3, p_1).$$

Функции выхода :

$$w_0 = \psi(s_0),$$

$$w_1 = \psi(s_1),$$

$$w_0 = \psi(s_2),$$

$$w_0 = \psi(s_3).$$

Таблица переходов и выхода заданного автомата Мура (табл.1.5).

Таблица 1.5

w_i	$\begin{matrix} p_i \\ s_i \end{matrix}$	p_1	p_2
w_0	s_0	s_0	s_1
w_1	s_1	s_2	s_3
w_0	s_2	s_2	s_3
w_0	s_3	s_3	s_0

Граф заданного автомата Мура (рис.1.7).

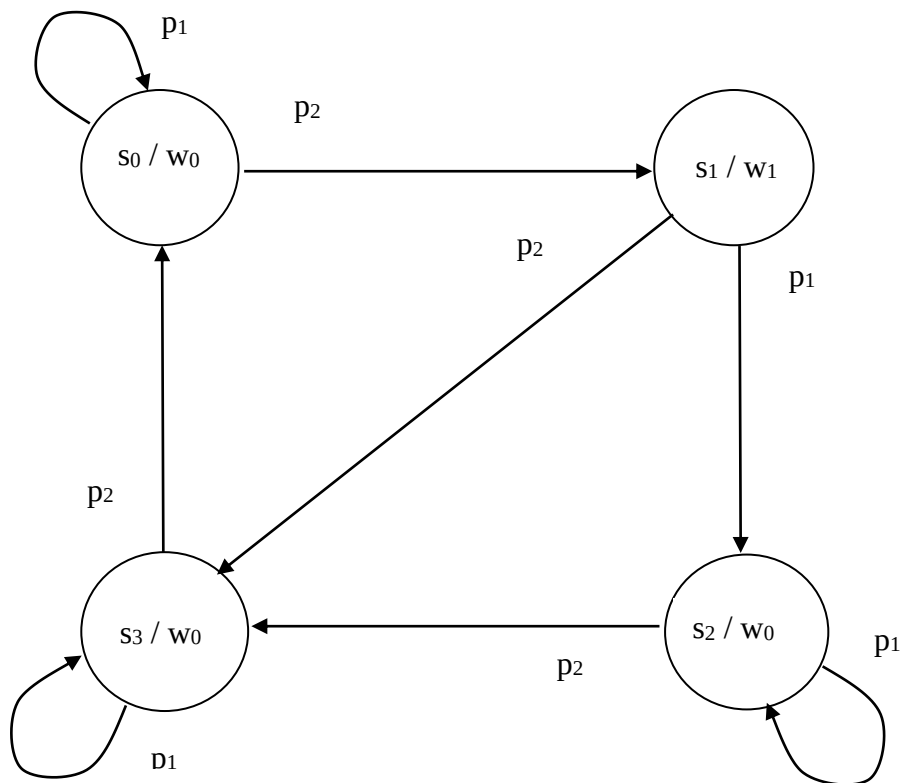


Рис.1.7

В табл.1.6 приводится моделирование работы автомата Мура, приведенного в примере, по преобразованию входного слова $p_1 p_2 p_2 p_1 p_2$. В процессе работы происходит следующая смена состояний автомата: $s_0 \rightarrow s_0 \rightarrow s_1 \rightarrow s_3 \rightarrow s_3 \rightarrow s_0$ и формируется следующая последовательность выходных символов $w_0 w_1 w_0 w_0 w_0$, т.е. работа автомата Мура состоит в преобразовании входного слова в выходное.

Таблица 1.6

t_k	t_0	t_1	t_2	t_3	t_4	t_5
S_i	S_0	S_0	S_1	S_3	S_3	S_0
P_j	p_1	p_2	p_2	p_1	p_2	
W_q	w_0^*	w_0	w_1	w_0	w_0	w_0

В момент t_0 значение w_0^* не учитывается, так как не является реакцией на входной символ, а определяется s_0 – начальным состоянием автомата Мура еще до подачи на вход первого символа входной последовательности.

Способы задания автомата Мили

Пусть заданы: P – входной алфавит, S – множество состояний, s_0 – начальное состояние автомата и W – выходной алфавит.

Функции перехода и функции выхода могут быть представлены тремя способами, как и в автомате Мура:

1. перечислением;
2. табличный;
3. графический.

При перечислении приводятся все функции перехода и выхода:

$$s_1 = \varphi(s_0, p_1),$$

$$w_1 = \psi(s_0, p_1),$$

$$s_2 = \varphi(s_1, p_1),$$

$$w_2 = \psi(s_1, p_1),$$

.....

.....

$$s_0 = \varphi(s_k, p_n).$$

$$w_0 = \psi(s_k, p_n).$$

Функция перехода задается как в автомате Мура. Отличие состоит в задании функции выхода.

В табличном способе строится таблица переходов (табл.1.7) и аналогичная ей таблица выходов (табл.1.8), на пересечении i – той строки и j – того столбца которой указывается выходной символ w_k , который выдает автомат при переходе из состояния s_i под воздействием входного символа p_j .

Таблица 1.7

$s_i \backslash p_j$	p_0	p_1	...	p_n
s_0	s_1	s_2	...	s_3
s_1	s_2	s_3	...	...
...	...	...	...	...
s_k	s_0	...	...	s_k

Таблица 1.8

$s_i \backslash p_j$	p_0	p_1	...	p_n
s_0	w_0	w_1	...	w_q
s_1	w_1	w_2	...	...
...	...	...	...	...
s_k	w_0	...	...	w_q

Эти таблицы аналогичны, поэтому обычно они объединяются, и получается совмещенная таблица переходов и выхода (табл.1.9).

Таблица 1.9

$s_i \backslash p_j$	p_0	p_1	...	p_n
s_0	s_1/w_0	s_2/w_1	...	s_3/w_q
s_1	s_2/w_1	s_3/w_2	...	...
...	...	...	...	...
s_k	s_0/w_0	...	...	s_k/w_q

В графическом способе каждому состоянию автомата ставится в соответствие вершина графа, которая помечается символом этого состояния s_i . Если из состояния s_i существует переход в состояние s_j под воздействием входного символа p_k , то вершины s_i и s_j соединяются дугой, исходящей из s_i , а сама дуга помечается символом p_k , под воздействием которого осуществляется данный переход, и выходным символом w_q , который вырабатывается при этом переходе (рис.1.8).

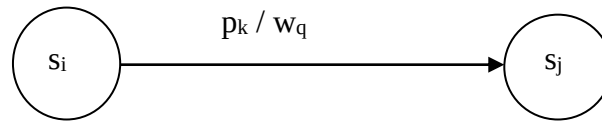


Рис.1.8

Пример автомат Мили.

Пусть $P = \{p_1, p_2\}$, $W = \{w_0, w_1\}$, $S = \{s_0, s_1, s_2\}$.

Функции перехода:

$$s_0 = \varphi(s_0, p_1),$$

$$s_0 = \varphi(s_2, p_2),$$

$$s_1 = \varphi(s_0, p_2),$$

$$s_1 = \varphi(s_1, p_1),$$

$$s_2 = \varphi(s_1, p_2),$$

$$s_2 = \varphi(s_2, p_1).$$

Функции выхода:

$$w_0 = \psi(s_0, p_1),$$

$$w_1 = \psi(s_0, p_2),$$

$$w_0 = \psi(s_1, p_1),$$

$$w_0 = \psi(s_0, p_2),$$

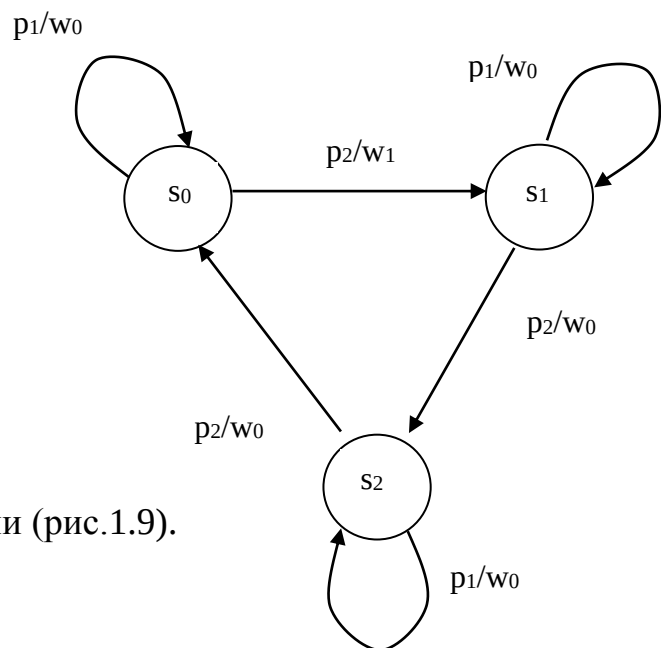
$$w_0 = \psi(s_2, p_1),$$

$$w_0 = \psi(s_2, p_2).$$

Таблица переходов и выхода для заданного автомата Мили (табл.1.10).

Таблица 1.10

P_j	p_1	p_2
S_i		
s_0	s_0/w_0	s_1/w_1
s_1	s_1/w_0	s_2/w_0
s_2	s_2/w_0	s_0/w_0



Граф заданного автомата Мили (рис.1.9).

Рис.1.9

В табл.1.11 приводится моделирование работы автомата Мили, приведенного в примере, по преобразованию входного слова $p_1p_2p_2p_1p_2$. В процессе работы происходит следующая смена состояний автомата: $s_0 \rightarrow s_0 \rightarrow s_1 \rightarrow s_2 \rightarrow s_2 \rightarrow s_0$ и формируется следующая последовательность выходных символов $w_0w_1w_0w_0w_0$, т.е. работа автомата Мили состоит в преобразовании входного слова в выходное.

Таблица 1.11

t_k	t_0	t_1	t_2	t_3	t_4	t_5
s_i	s_0	s_0	s_1	s_2	s_2	s_0
p_j	p_1	p_2	p_2	p_1	p_2	
w_q		w_0	w_1	w_0	w_0	w_0

При сравнении работы автомата Мура (табл.1.5) и работы автомата Мили (табл.1.10) видно, что одно и то же входное слово $p_1p_2p_2p_1p_2$ оба автомата перерабатывают в одно и то же выходное слово $w_0w_1w_0w_0w_0$. Разница в работе автомата Мура и автомата Мили в том, что в автомате Мура выходной символ можно считать сразу после установки его в начальное состояние, но при этом надо учитывать, что этот символ не является реакцией на входной символ, а определяется только начальным состоянием автомата Мура.

В автомате Мили выходной символ можно считать после установки его в начальное состояние и подачи на его вход первого входного символа.

1.5. ЭКВИВАЛЕНТНОСТЬ АВТОМАТОВ МИЛИ И МУРА

Два автомата, у которых входные и выходные алфавиты совпадают или равнозначны, называют *эквивалентными*, если любое входное слово оба автомата перерабатывают в совпадающие выходные слова, при условии, что перед началом работы оба автомата находились в начальном состоянии.

Построение автомата Мили, эквивалентного заданному автомату Мура.

Пусть задан автомат Мура: $A^0 = \langle P^0, S^0, s_0^0, \varphi^0, W^0, \psi^0 \rangle$.

Найти автомат Мили: $A = \langle P, S, s_0, \varphi, W, \psi \rangle$.

Из определения эквивалентности следует: $P = P^0, W = W^0, s_0 = s_0^0$.

Как в автомате Мура, так и в автомате Мили следующее состояние зависит от текущего состояния и текущего входного символа:

в автомате Мура – $s_k^0(t+1) = \varphi^0(s_i^0(t), p_j^0(t))$;

в автомате Мили – $s_k(t+1) = \varphi(s_i(t), p_j(t))$.

Из этого следует, что функция перехода автомата Мили совпадает с функцией перехода автомата Мура при одном и том же входном символе. При этом число состояний и функций переходов не изменится т.е. :

$$S = S^o \text{ и } \varphi(s_i, p_j) = \varphi^o(s_i^o, p_j^o).$$

В автомате Мили выходной символ $w_k(t+1)$ определяется текущим состоянием $s_i(t)$ и входным символом $p_j(t)$:

$$w_k(t+1) = \psi(s_i(t), p_j(t))$$

В автомате Мура выходной символ $w_k^o(t+1)$ определяется следующим состоянием $s_k^o(t+1)$:

$$w_k^o(t+1) = \psi^o(s_k^o(t+1)), \text{ а это состояние: } s_k^o(t+1) = \varphi^o(s_i^o(t), p_j^o(t)).$$

Если подставить это выражение для $s^o(t+1)$ в функцию выхода автомата Мура, получим его определение через старое состояние:

$w_k^o(t+1) = \psi^o(s_k^o(t+1)) = \psi^o(\varphi^o(s_i^o(t), p_j^o(t)))$, а так как $\varphi(s_i, p_j) = \varphi^o(s_i^o, p_j^o)$, то для обеспечения условий эквивалентности необходимо выходные символы автомата Мили, получаемые на переходах в новые состояния, сделать равными выходным символам автомата Мура этих состояний и в результате получаем: $w_k(t+1) = \psi(s_i(t), p_j(t))$.

Преобразование автомата Мура в автомат Мили удобно производить в графическом представлении автоматов. Для этого необходимо выходные символы, которыми помечены вершины графа автомата, перенести на все дуги, входящие в каждую вершину.

Например, пусть задан граф автомата Мура (рис.1.10).

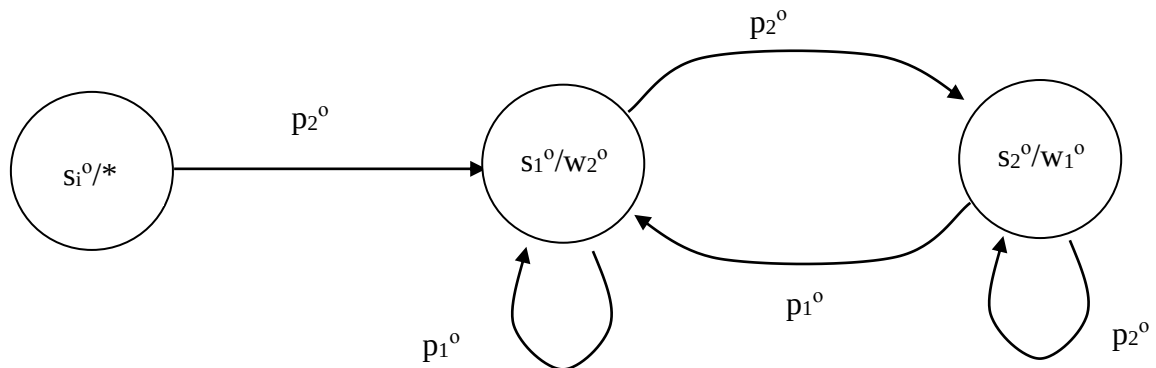


Рис.1.10

После переноса выходного символа из каждой вершины на каждую дугу, входящую в эту вершину, получим граф автомата Мили (рис.1.11).

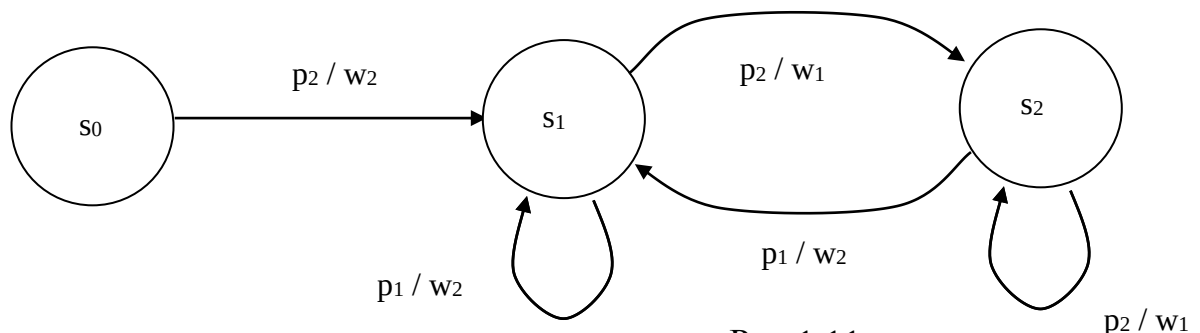


Рис.1.11

Построение автомата Мура, эквивалентного заданному автомату Мили.

Пусть задан автомат Мили: $A = \langle P, S, s_0, \varphi, W, \psi \rangle$.

Найти автомат Мура: $A^0 = \langle P^0, S^0, s_0^0, \varphi^0, W^0, \psi^0 \rangle$.

Из определения эквивалентности следует: $P^0 = P, W^0 = W, s_0^0 = s_0$.

Как в автомате Мили, так и в автомате Мура следующее состояние зависит от текущего состояния и текущего входного символа:

в автомате Мили – $s_k(t+1) = \varphi(s_i(t), p_j(t))$;

в автомате Мура – $s_k^0(t+1) = \varphi^0(s_i^0(t), p_j^0(t))$.

Из этого следует, что функция перехода автомата Мура аналогична функции перехода автомата Мили при одном и том же входном символе. Однако так как в автомате Мура выходной символ $w_k^0(t+1)$ определяется следующим состоянием $s_k^0(t+1)$, то каждому состоянию может соответствовать только один выходной символ. Таким образом, если в исходном автомате Мили существуют переходы в одно и то же состояние с выдачей различных выходных символов, то в эквивалентном ему автомате Мура число состояний и соответственно число функций переходов увеличится т.е.:

$[S^0] \geq [S]$ и $\varphi^0(s_i^0, p_j^0) \neq \varphi(s_i, p_j)$.

Преобразование автомата Мили в автомат Мура можно производить при графическом представлении автоматов. Для этого необходимо выполнить действия, обратные действиям при преобразовании автомата Мура в автомат Мили, т.е. выходной символ, которым помечена дуга графа автомата, перенести в вершину, в которую эта дуга входит.

Например, для фрагмента графа автомата Мили (рис.1.12), сделав такой перенос, получим фрагмент графа автомата Мура (рис.1.13).

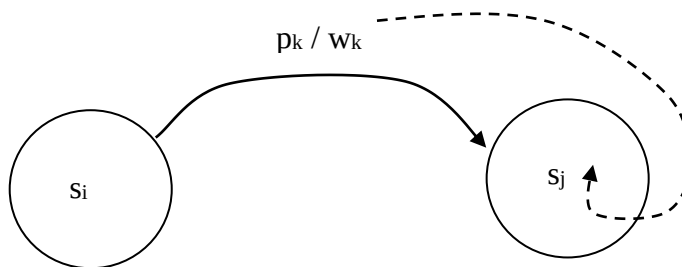


Рис.1.12

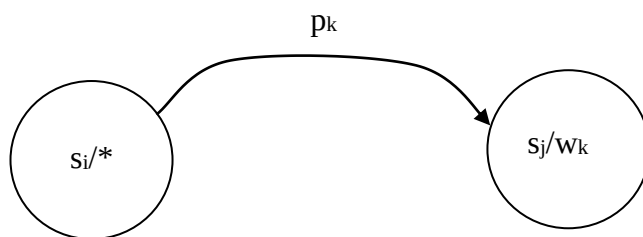


Рис.1.13

Для фрагмента графа автомата Мили на рис.1.14 такой перенос осуществить нельзя, так как на каждом переходе происходит выдача различных выходных символов. В этом случае состояние s_m необходимо расщепить (продублировать) на такое количество состояний, сколько различных входных символов имеется на всех входных ребрах этого состояния, и сопоставить каждому из них свой выходной символ (рис.1.15).

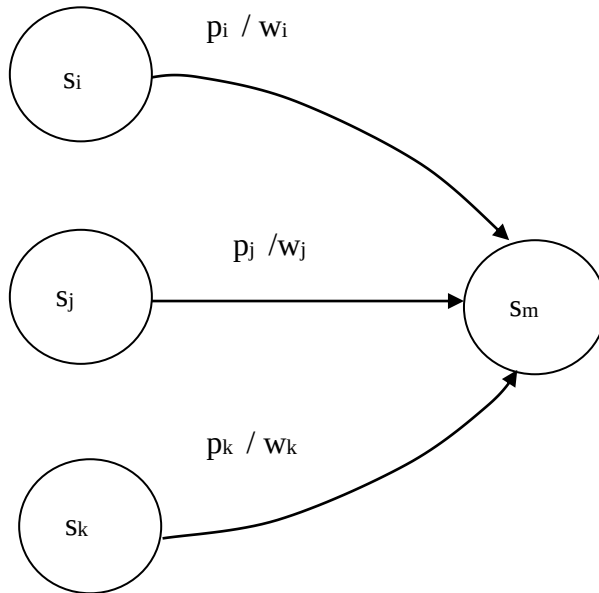


Рис.1.14

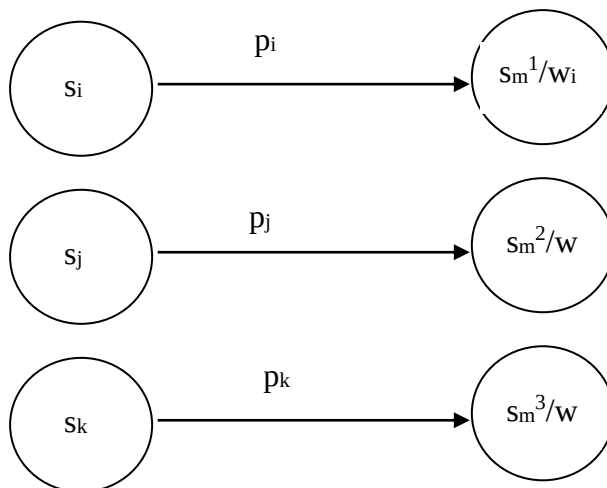
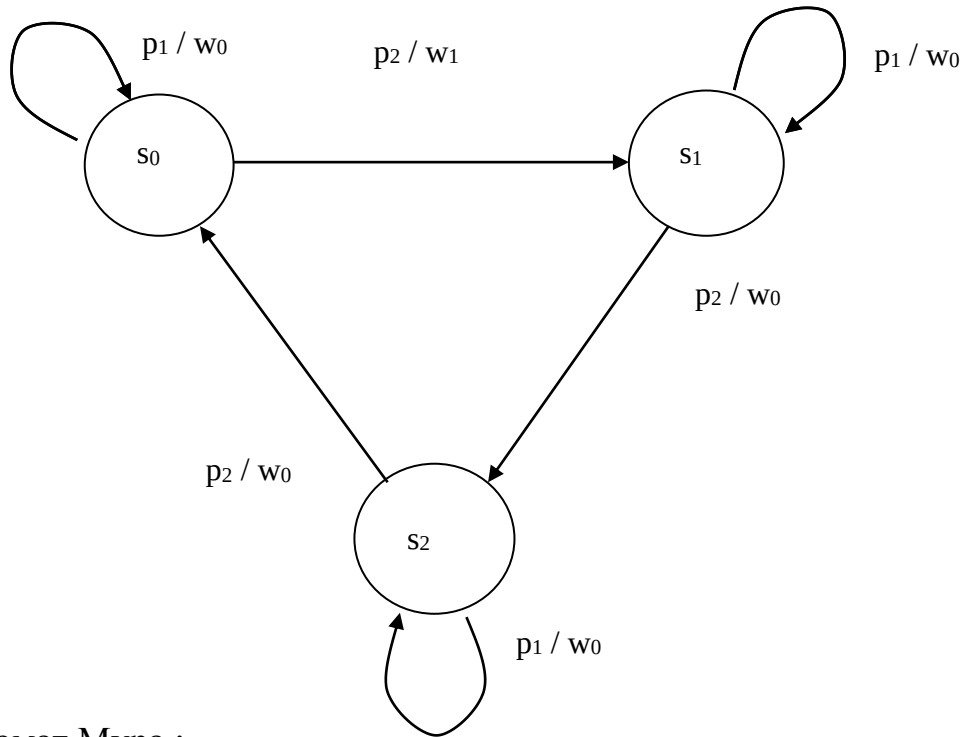


Рис.1.15

В общем случае при переходе к автомату Мура число состояний может увеличиваться и если одно и то же устройство описывается разными моделями автоматов, то в модели автомата Мура может быть больше число состояний.

Рассмотрим переход от автомата Мили, представленного ранее (рис. 1.9), к модели автомата Мура (рис. 1.16).

Автомат Мили :



Автомат Мура :

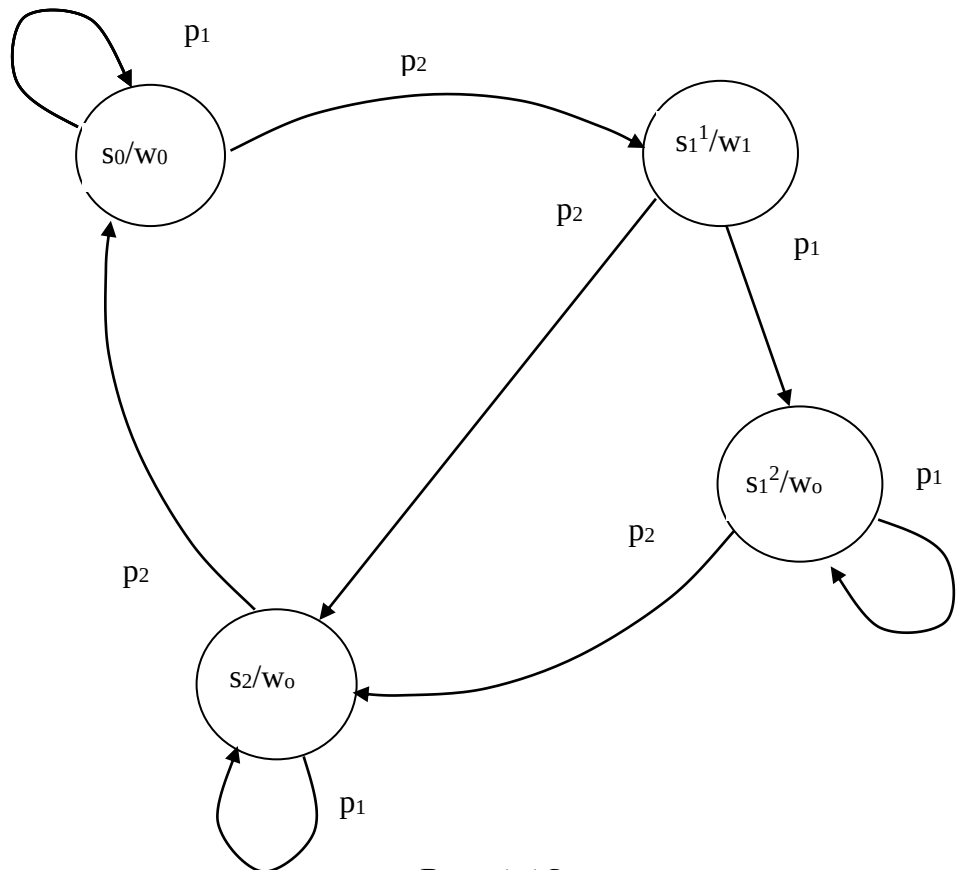


Рис. 1.16

Состояние s_1 автомата Мили в процессе перехода было расщеплено на s_1^1 и s_1^2 автомата Мура. Из сравнения автомата Мура (рис. 1.9) с

автоматом на рис. 1.16 видно, что это один тот же автомат, для которого ранее было проведено моделирование, и результат работы (табл. 1.6) совпадает с результатом работы автомата Мили (табл. 1.11), т.е. эти автоматы эквивалентны.

Частичные или не полностью определенные автоматы.

Пусть $P = \{p_0, p_1, \dots, p_n\}$ – входной алфавит;

$\tilde{P}^*$ – множество всех слов в алфавите P ;

$\tilde{P}_g^*$ – множество допустимых слов ($\tilde{P}_g^* \in \tilde{P}^*$). Это множество входных слов, для которых определено множество выходных слов;

$\tilde{P}_z^* = \tilde{P} \setminus \tilde{P}_g^*$ – множество запрещенных слов.

Автомат называется *частичным или не полностью определенным*, если множество запрещенных слов не пусто: $\tilde{P}_z^* \neq \emptyset$.

В таблице переходов и выхода такого автомата будут прочерки, означающие отсутствие переходов.

Пример: таблица переходов и выхода (табл. 1.12) и граф (рис. 1.17) частичного автомата Мили.

Таблица 1.12

$S_i \backslash p_j$	p_1	p_2	p_3
S_0	S_1/w_1	---	---
S_1	---	S_2/w_1	S_2/w_2
S_2	---	S_0/w_3	S_3/w_1
S_3	---	S_0/w_1	S_0/w_3

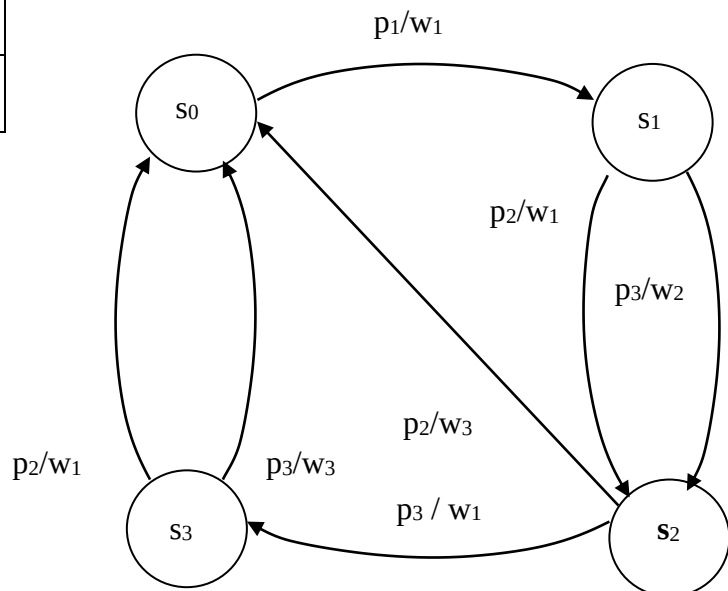


Рис. 1.17

В графе частичного автомата Мили будут отсутствовать дуги, соответствующие отсутствующим переходам.

2. КАНОНИЧЕСКИЙ МЕТОД СИНТЕЗА КОНЕЧНЫХ АВТОМАТОВ

Канонический метод синтеза конечных автоматов состоит из двух этапов:

- этап абстрактного синтеза;
- этап структурного синтеза.

2.1. АБСТРАКТНЫЙ СИНТЕЗ КОНЕЧНЫХ АВТОМАТОВ

Исходными данными в каноническом методе синтеза на абстрактном этапе является таблица соответствия между входными и выходными словами.

Результатом абстрактного этапа является абстрактный автомат, представленный в виде графа или таблицы переходов и выхода.

Этап абстрактного синтеза состоит из двух подэтапов:

1. Построение по алфавитному оператору абстрактного автомата.
2. Минимизация числа состояний абстрактного автомата.

Таблица, устанавливающая соответствие между входными и выходными словами, которые перерабатывает автомат, называется алфавитным оператором.

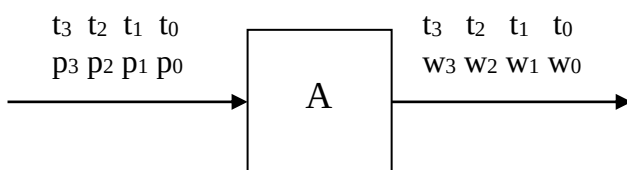


Рис. 2.1

Индексы (рис. 2.1) указывают на соответствие входных и выходных символов автоматным тактам, а не на различие между буквами.

Пусть $P = \{0, 1\}$, $W = \{0, 1\}$.

В табл. 2.1 приведен пример алфавитного оператора, где индексы соответствуют автоматным тактам и задают последовательность считывания и выдачи символов.

Таблица 2.1

p ₀	p ₁	p ₂	p ₃	w ₀	w ₁	w ₂	w ₃
0	0	0	0	0	0	0	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	0	0
0	0	1	0	1	0	0	0

Алфавитный оператор называется автоматным, если он удовлетворяет следующим условиям:

- 1) однозначно отображает входные слова в выходные;
- 2) удовлетворяет условию полноты, т.е. любой начальный отрезок $\tilde{p}$ допустимого слова также допустим ($\tilde{p} \in P_g^*$);
- 3) сохраняет длину преобразуемого слова;
- 4) осуществляет преобразование совпадающих начальных отрезков входных слов в совпадающие начальные отрезки соответствующих выходных слов.

Если алфавитный оператор является автоматным, то можно построить реализующий его конечный автомат.

Для преобразования неавтоматного алфавитного оператора к автоматному виду используются две операции: выравнивание и пополнение.

Операция *выравнивание* используется, если нарушено третье условие автоматности оператора и состоит в приписывании специального пустого символа « e » справа к входному или слева к выходному слову, при этом этот пустой символ добавляется во входной и выходной алфавиты.

В табл. 2.2 приведен пример применения операция *выравнивание* к алфавитному оператору, преобразующему двухбуквенные слова в однобуквенные.

Таблица 2. 2

p_0	p_1	w_0	w_1
0	0	e	0
0	1	e	1
1	0	e	0
1	1	e	1

Операция *пополнение* используется, если нарушено четвертое условие автоматности, и состоит в приписывании пустого символа « e » справа к входному слову и одновременно слева к выходному слову.

В табл. 2.3 приведен пример применения операция *выравнивание* к алфавитному оператору, преобразующему двухбуквенные слова в двухбуквенные.

Таблица 2. 3

p_0	p_1	p_1	w_0	w_1	w_2
0	0	e	e	0	1
0	1	e	e	1	0
1	0	e	e	1	1
1	1	e	e	0	1

Первоначально одинаковые начальные отрезки из одной буквы 0 в первом и втором входном слове преобразуются в 0 в первом и в 1 во втором выходном слове, т.е. нарушено четвертое условие автоматности, поэтому к этим входным словам справа и к соответствующим выходным словам слева дописывается пустой символ « е ». После этого одинаковые начальные отрезки первого и второго входного слова преобразуются в одинаковые начальные отрезки выходных слов из буквы « е ». То же самое необходимо проделать с третьим и четвертым словом.

Алфавитные операторы могут быть заданы двумя способами: табличным и графическим.

Табличный способ устанавливает соответствие между входными и выходными словами в форме таблицы.

В графическом способе оператор представляется в виде дерева нагруженного входными и выходными словами. Дерево это граф без циклов. Путь от корня (начальной вершины) к листу (конечной вершине) дерева соответствует одному входному слову. Число ярусов дерева соответствует длине входного слова.

Построение дерева входных последовательностей.

Пусть $P = \{p_0, p_1, \dots, p_n\}$ – входной алфавит.

Алгоритм построения дерева состоит из выполнения следующих шагов:

- 1) фиксируется начальная вершина, которая называется *корнем* дерева;
- 2) из вершины проводятся n дуг, каждая из которых помечается буквой входного алфавита;
- 3) на концах дуг строится n вершин;
- 4) из каждой построенной вершины проводятся n дуг и так далее;
- 5) последняя висячая вершина (*лист*) – называется конечной вершиной.

Для построения автомата по автоматному алфавитному оператору необходимо выполнить следующие действия :

- 1) построить дерево входных последовательностей;
- 2) отметить все вершины дерева символами состояний автомата s_i , при этом корень пометить символом начального состояния s_0 , а все висячие вершины (листья) символами конечного состояния s_k ;
- 3.1) для получения графа автомата Мура вершины дерева отметить символами соответствующих выходных слов;
- 3.2) для получения графа автомата Мили отметить дуги символами соответствующих выходных слов;
- 4) все конечные вершины совместить в одну.

В результате выполнения этого подэтапа получается граф полностью определенного абстрактного автомата.

Минимизация числа состояний абстрактного автомата, заданного графом переходов .

Уменьшение числа состояний полностью определенного абстрактного автомата можно осуществить за счет объединения эквивалентных состояний.

Два состояния автомата называются эквивалентными, если при переходе из них по одинаковым входным символам в одно и то же состояние происходит выдача одинаковых выходных символов.

На рис. 2.2 представлен фрагмент графа автомата Мили с двумя эквивалентными состояниями s_i и s_j , что обозначается: $s_i \equiv s_j$.

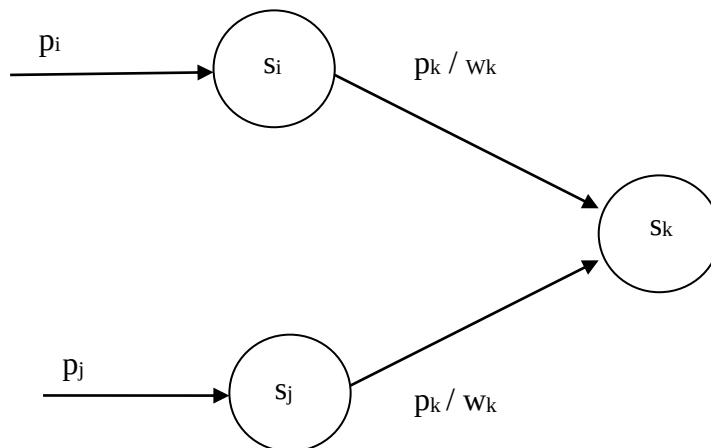


Рис. 2.2

После объединения этих эквивалентных состояний в одно, обозначенное символом s_{ij} , получается фрагмент графа автомата Мили (рис. 2.3) с меньшим числом состояний.

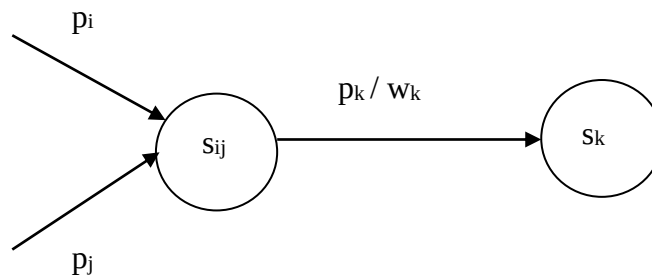


Рис. 2.3

Автоматом многократного действия называется автомат, который после переработки входного слова будет готов к приему следующего.

Для получения автомата многократного действия необходимо его начальную вершину совместить с конечной.

Пример построения абстрактного автомата Мили по автоматному алфавитному оператору, заданного таблицей (табл. 2.4).

Таблица 2. 4

p_0	p_1	p_2	w_0	w_1	w_2
0	0	0	0	0	1
0	0	1	0	0	0
0	1	0	0	0	1
0	1	1	0	0	0
1	0	0	0	0	1
1	0	1	0	0	0
1	1	0	0	0	1
1	1	1	0	0	0

Из таблицы определяется входной и выходной алфавит:
 $P = \{0, 1\}$, $W = \{0, 1\}$.

Строится дерево входных последовательностей (рис. 2.4).

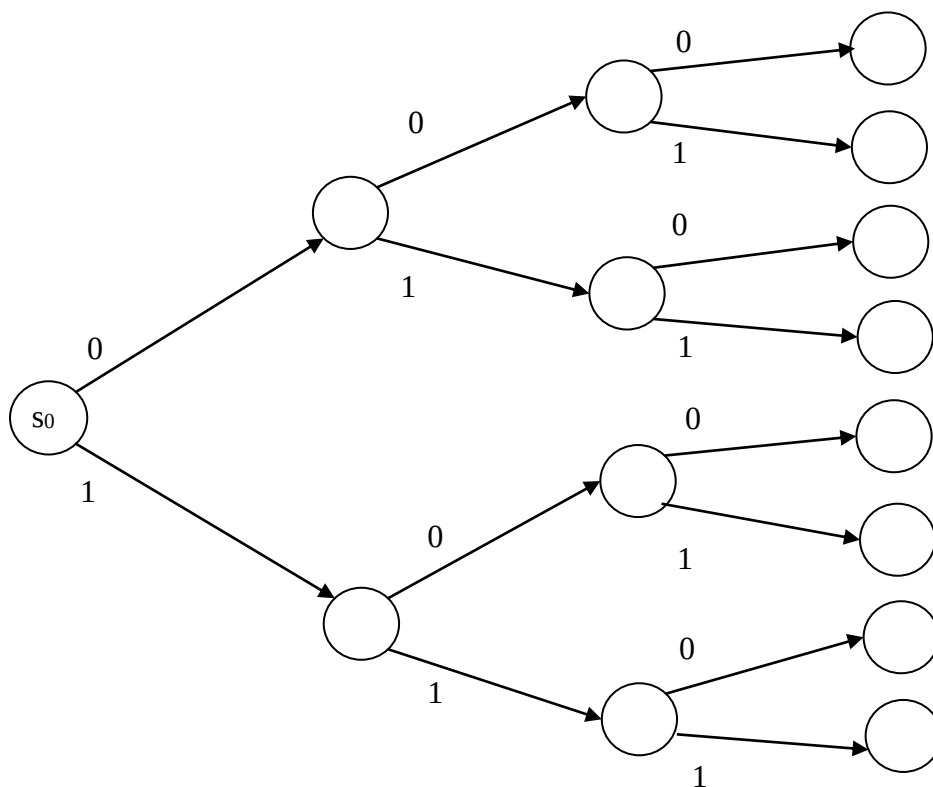


Рис.2.4

Дерево входных последовательностей с вершинами, размеченными символами состояний (рис. 2.5), формирует следующее множество состояний: $S = \{s_0, s_1, s_2, s_3, s_4, s_5, s_6, s_k\}$.

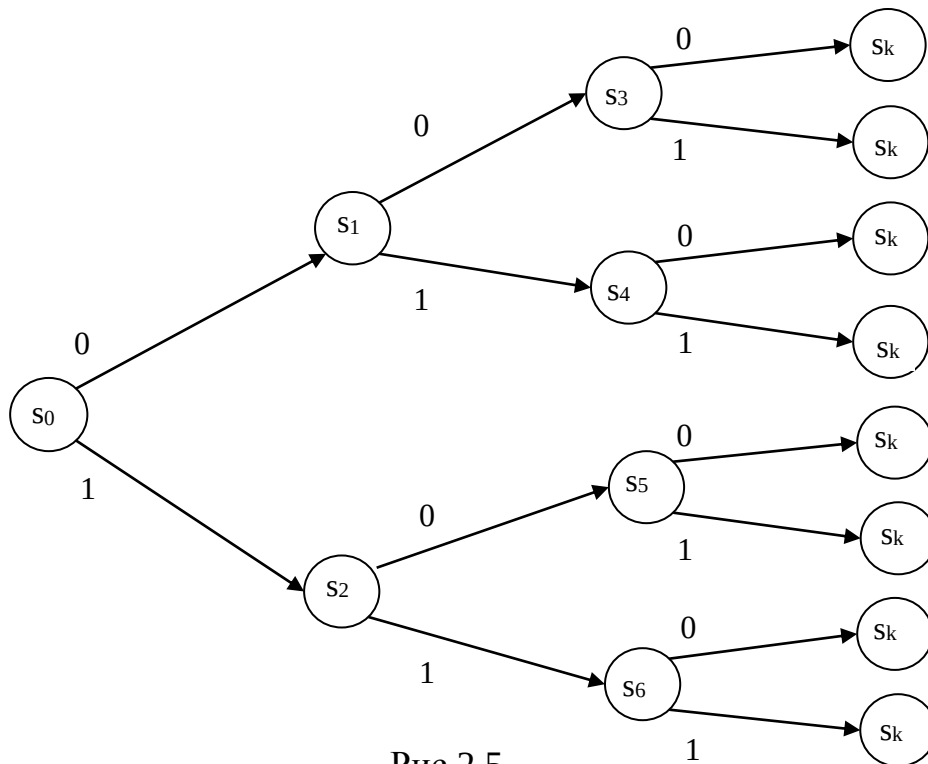


Рис.2.5

Дерево с дугами, размеченными соответствующими выходными символами (рис. 2.6).

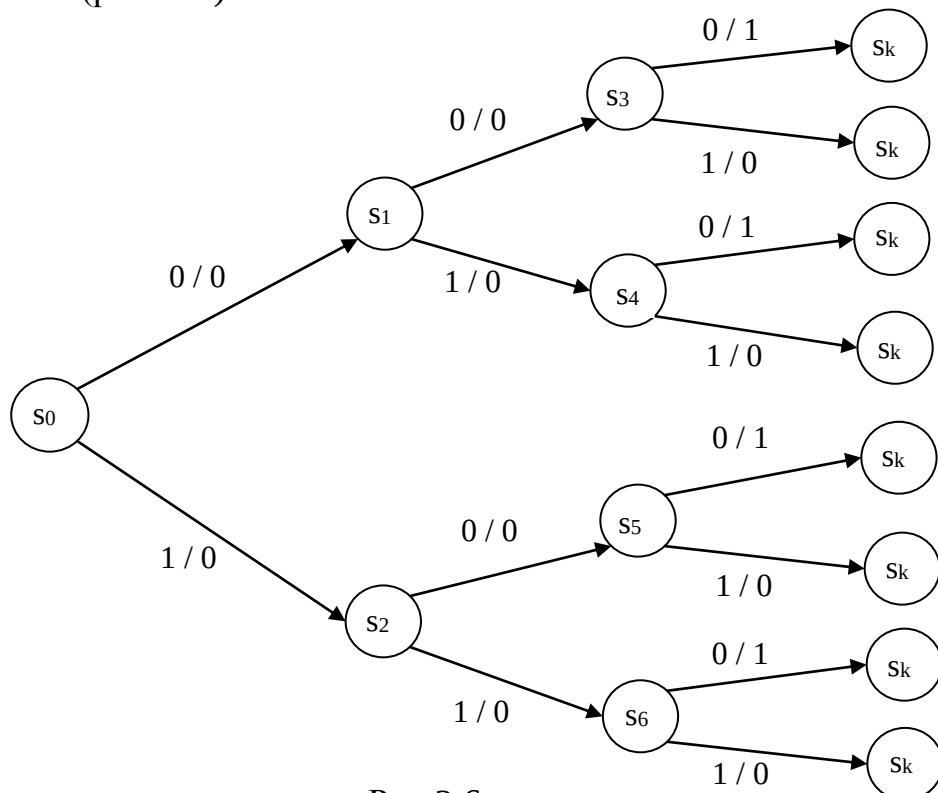


Рис.2.6

После объединения в одну всех вершин, помеченных символом s_k , получается граф автомата Мили (рис. 2.7).

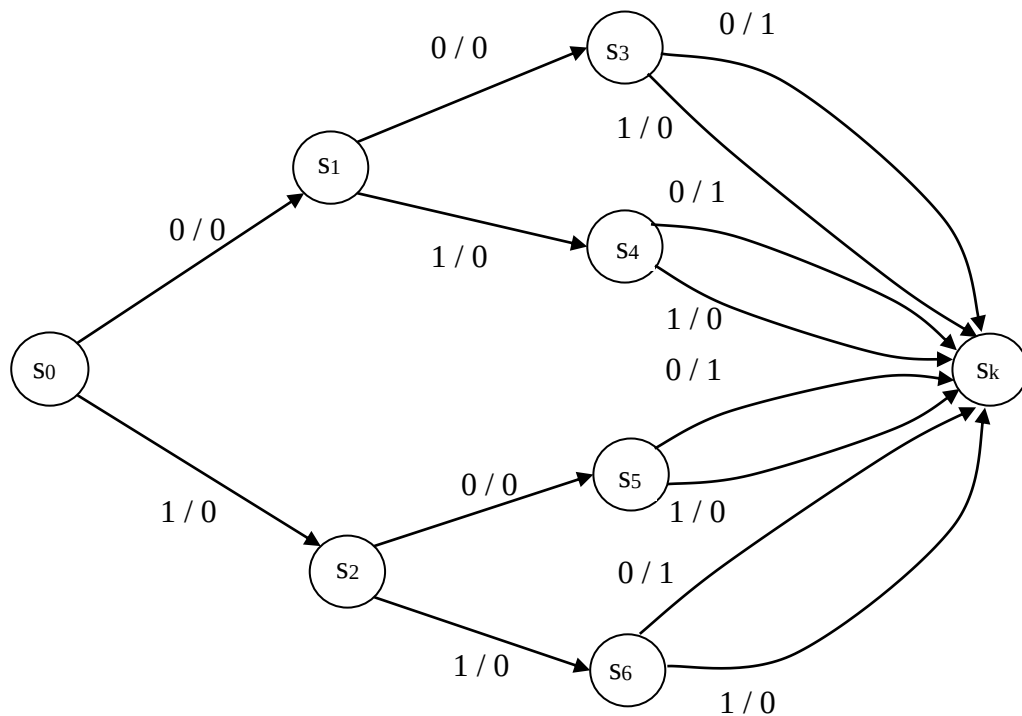


Рис.2.7

Вторым этапом абстрактного синтеза является минимизация числа состояний. Из анализа переходов в конечное состояние s_k видно, что существуют две пары эквивалентных состояний $s_3 \equiv s_4$ и $s_5 \equiv s_6$. После объединения этих состояний получается граф автомата Мили (рис. 2.8).

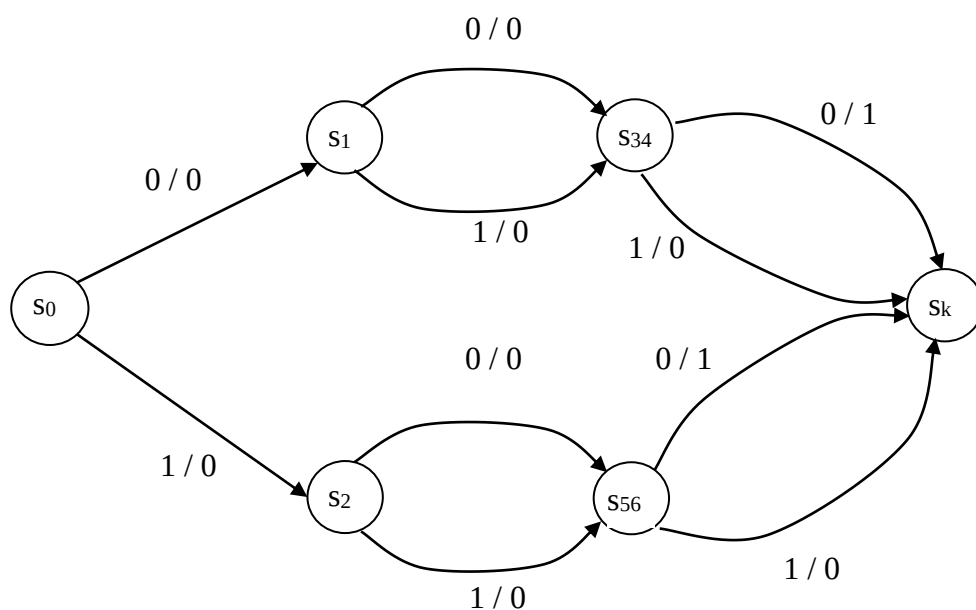


Рис.2.8

Из анализа переходов в конечное состояние s_k на рис.2.8 видно, что новые состояния s_{34} и s_{56} также эквивалентны, т.е. $s_{34} \equiv s_{56}$. После их объединения получается граф автомата Мили (рис. 2.9).

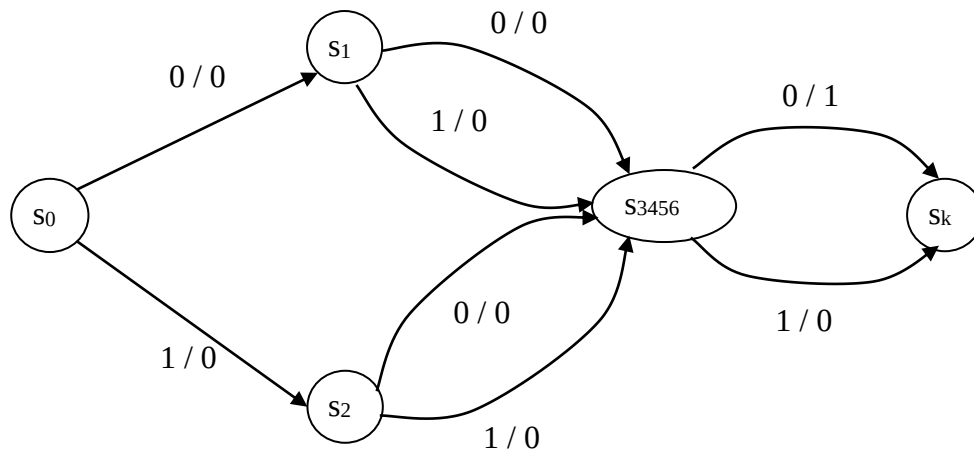


Рис.2.9

Из анализа переходов в состояние s_{3456} видно, что $s_1 \equiv s_2$ и после их объединения получается граф автомата Мили (рис. 2.10).

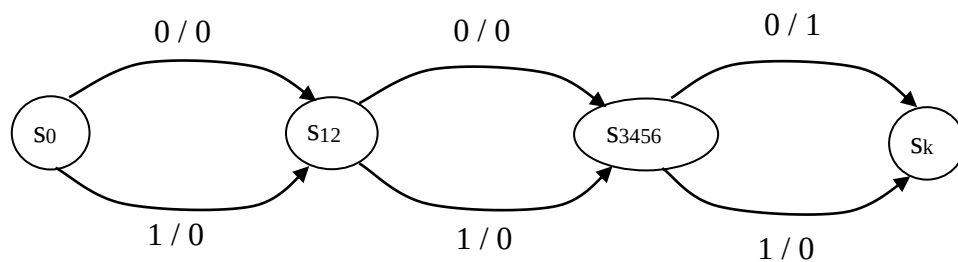


Рис.2.10

Для получения автомата многократного действия совмещаются состояния s_0 и s_k и после их переобозначения получается граф автомата Мили (рис. 2.11).

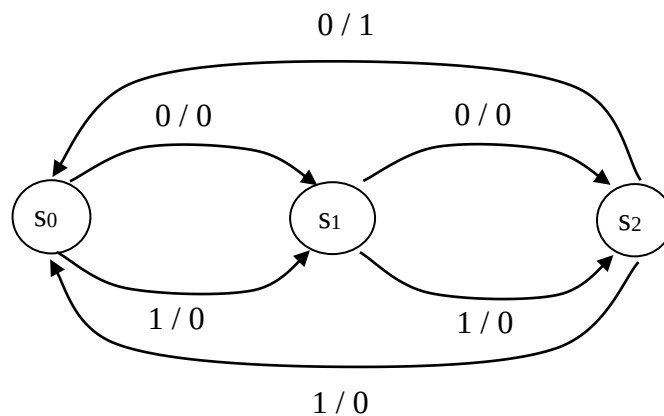


Рис. 2.11

Графу автомата Мили (рис. 2.11) соответствует таблица переходов и выхода (табл. 2.5).

Таблица 2.5

$s_j \backslash p_i$	0	1
s_0	$s_1/0$	$s_1/0$
s_1	$s_2/0$	$s_2/0$
s_2	$s_0/1$	$s_0/0$

Для проверки работы полученного автомата промоделируем его работу по переработке входного слова: 1 1 0. Результат работы приведен в табл. 2.6.

Таблица 2.6

t_k	t_0	t_1	t_2	
s_i	s_0	s_1	s_2	s_0
p_j	1	1	0	
w_m		0	0	1

Таким образом, входное слово: 1 1 0 перерабатывается в выходное слова: 0 0 1, что соответствует заданному алфавитному оператору.

2.2. МИНИМИЗАЦИЯ ЧИСЛА СОСТОЯНИЙ ПОЛНОСТЬЮ ОПРЕДЕЛЕННОГО АВТОМАТА

В примере была рассмотрена минимизация автомата Мили на его графе, но это трудно проделать при большом числе состояний.

Пусть полностью определенный автомат A , имеющий k состояний, задан таблицей переходов и выхода. Требуется построить эквивалентный ему автомат A^* с числом состояний k^* , причем $k^* \leq k$.

Минимизация числа состояний полностью определенного автомата осуществляется за счет объединения эквивалентных состояний.

Два состояния s' и s'' называются *эквивалентными*, если любую входную последовательность автомат перерабатывает в одинаковые выходные последовательности, независимо от того, какой из этих двух состояний s' или s'' выбраны в качестве начального т.е. $s' \equiv s''$.

Отношение эквивалентности обладает следующими свойствами:

- 1) рефлексивность, т.е. $s' \equiv s'$;
- 2) симметричность, т.е. если $s' \equiv s''$, то из этого следует $s'' \equiv s'$;
- 3) транзитивность, т.е. если $s' \equiv s_k$ и $s'' \equiv s_k$, то из этого следует, что $s' \equiv s''$.

Отношение эквивалентности разбивает все множество состояний S на классы эквивалентности: $S = C_1 \cup C_2 \cup C_3 \cup \dots \cup C_q$.

Класс эквивалентности $C_i \in S$ – это подмножество состояний, попарно эквивалентных между собой, причем $C_i \cap C_j = \emptyset$, если $i \neq j$, т.е. пересечение любых двух классов эквивалентности представляет собой пустое множество и следовательно каждое состояние может входить только в один класс эквивалентности.

На рис. 2.12 приведен тривиальный пример эквивалентных состояний $s' \equiv s''$ и их объединения, который был использован ранее при переходе от алфавитного оператора к графу автомата.

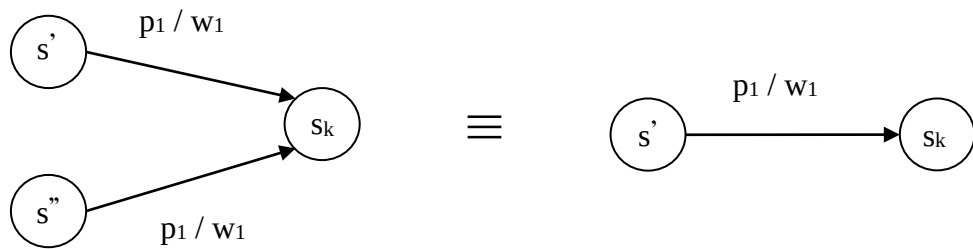


Рис. 2.12

На рис. 2.13 приведен пример фрагмента графа автомата, в котором эквивалентность состояний s' и s'' не является столь очевидной.

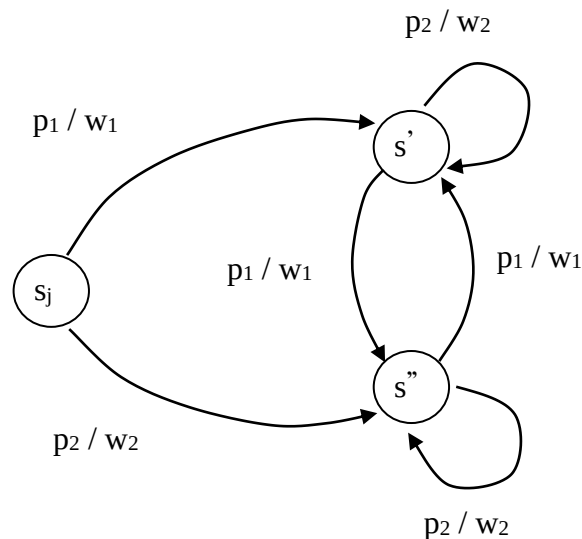


Рис. 2.13

На рис. 2.14 приведен фрагмент графа после объединения s' и s'' .

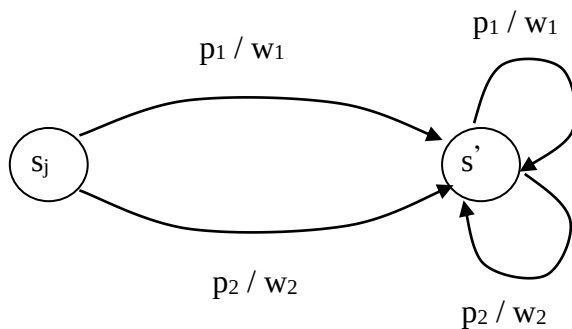


Рис. 2.14

Два необходимых условия эквивалентности состояний автомата Мили

Два состояния s' и s'' являются эквивалентными, если:

- при переходе из этих состояний под воздействием любого одинакового входного символа p_k вырабатываются одинаковые выходные символы, т.е: $\psi'(s', p_k) = \psi''(s'', p_k)$;
- при переходе из этих состояний под воздействием любого одинакового входного символа p_k переход осуществляется в одно и то же или в эквивалентные состояния, т.е. если $s_i = \phi'(s', p_k)$ и $s_j = \phi''(s'', p_k)$, то следует, что $s_i \equiv s_j$.

Алгоритм минимизации числа состояний полностью определенного автомата с помощью треугольной матрицы.

Задан таблицей переходов и выхода полностью определенный автомат Мили, имеющий k – состояний.

Алгоритм состоит из выполнения следующих шагов:

1. Строится треугольная матрица без диагональных элементов, столбцы которой нумеруются слева направо индексами состояний с 1 до $k-1$, а строки матрицы сверху вниз с 2 до k .

2. Любой клетке матрицы с координатами i и j соответствует пара состояний s_i и s_j . Последовательно просматривая таблицу переходов матрица заполняется следующим образом:

- а) если для пары s_i и s_j невыполнимо первое условие эквивалентности, то в клетку ставят «X», это означает, что эти состояния заведомо неэквивалентны;
- б) если для пары состояний s_i и s_j выполняется первое условие, но еще неизвестно, выполняется ли второе, так как переходы осуществляется в разные состояния, то в клетку записывают пары номеров состояний, от эквивалентности которых зависит эквивалентность рассматриваемой пары;
- с) если для пары состояний s_i и s_j сразу выполняются оба условия, то клетка оставляется пустой, и это означает, что эти состояния заведомо эквивалентны.

3. Заполненная треугольная матрица, последовательно просматривается по клеткам, содержащим номера пар состояний, и, если известно, что эти номера соответствуют неэквивалентным состояниям, то в этой клетке ставят «X». Просмотр матрицы продолжается до тех пор, пока не перестанут появляться новые пары неэквивалентных состояний;

4. Из треугольной матрицы выписываются все пары эквивалентных состояний (где не содержится «X») и из них образуются классы эквивалентности.

5. Каждому классу эквивалентности ставится в соответствие символ нового состояния и переписывается исходная таблица переходов и выхода.

Пример минимизации автомата Мили, заданного таблицей переходов и выхода (табл. 2.7).

$$P = \{a, b, c\}, W = \{1, 2\}, S = \{s_1, s_2, s_3, s_4, s_5, s_6, s_7, s_8\}.$$

В таблице переходов и выхода (табл. 2.7) для наглядности вместо символов состояний указаны только их индексы.

Таблица 2.7

$P_k \backslash S_i$	a	b	c
1	4 / 1	2 / 2	5 / 1
2	5 / 2	1 / 1	4 / 2
3	3 / 2	5 / 1	4 / 2
4	5 / 1	8 / 2	4 / 2
5	7 / 1	2 / 2	1 / 1
6	1 / 1	3 / 2	4 / 2
7	5 / 1	3 / 2	7 / 2
8	3 / 2	5 / 1	6 / 2

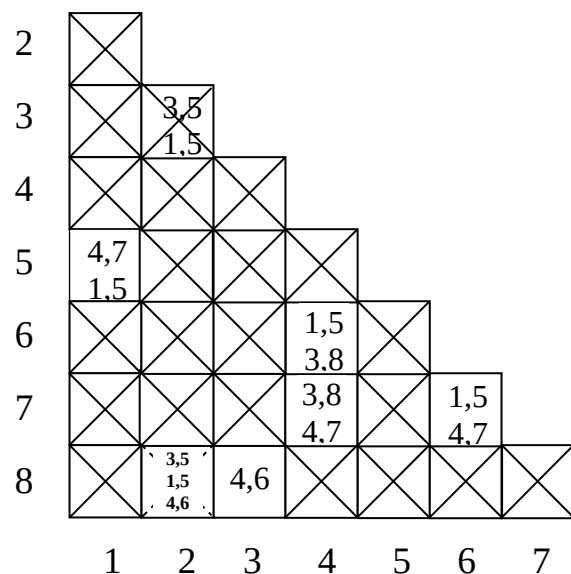


Рис. 2.15

На рис. 2.15 приведена треугольная матрица, заполненная по изложенному выше алгоритму. Клетки, не содержащие «X» соответствуют следующим парам эквивалентных состояний: $1 \equiv 5$, $3 \equiv 8$, $4 \equiv 6$, $4 \equiv 7$, $6 \equiv 7$ и из них можно образовать следующие классы эквивалентности:

$$C_1 = \{1, 5\}, C_2 = \{2\}, C_3 = \{3, 8\}, C_4 = \{4, 6, 7\}.$$

Каждому классу сопоставляется символ нового состояния:

$C_1 - s'_1$, $C_2 - s'_2$, $C_3 - s'_3$, $C_4 - s'_4$. Далее в исходной таблице переходов символ старого состояния заменяется на символ нового состояния, который соответствует классу, в который это старое состояние входит. В результате получается таблица переходов и выхода (табл. 2.8).

После объединения в ней одинаковых строк получается таблица переходов и выхода минимального автомата Мили (табл. 2.9), с числом состояний $k^* = 4$ вместо $k = 8$ в исходном автомате.

Таблица 2.8

$S_i \backslash p_k$	a	b	c
1'	4'/1	2'/2	1'/1
2'	1'/2	1'/1	4'/2
3'	3'/2	1'/1	4'/2
4'	1'/1	3'/2	4'/2
1'	4'/1	2'/2	1'/1
4'	1'/1	3'/2	4'/2
4'	1'/1	3'/2	4'/2
3'	3'/2	1'/1	4'/2

Таблица 2.9

$S_i \backslash p_k$	a	b	c
1'	4'/1	2'/2	1'/1
2'	1'/2	1'/1	4'/2
3'	3'/2	1'/1	4'/2
4'	1'/1	3'/2	4'/2

2.3. МИНИМИЗАЦИЯ ЧИСЛА СОСТОЯНИЙ ЧАСТИЧНОГО АВТОМАТА

Два частичных автомата называются совместимыми (слабо эквивалентными), если:

- 1) они имеют одинаковое множество допустимых входных слов;
- 2) любое допустимое входное слово оба автомата перерабатывают в непротиворечивые выходные слова при условии, что перед началом работы оба автомата находились в начальном состоянии.

Два выходных слова двух частичных автоматов A и A^* называются непротиворечивыми, если все символы, которые определены в выходном слове автомата A совпадают со всеми соответствующими символами, которые определены в выходном слове автомата A^* .

На рис. 2.16 приведен пример непротиворечивых выходных слов.

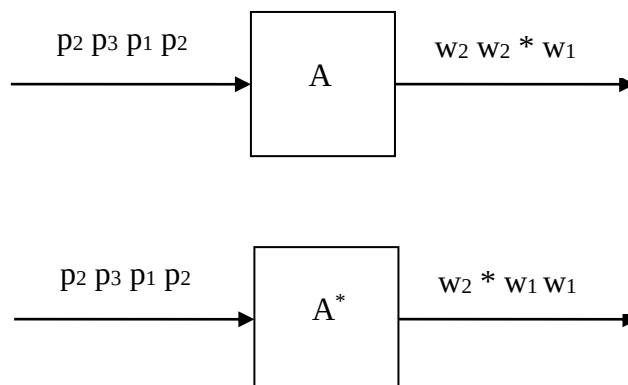


Рис. 2.16

Два состояния s' и s'' частичного автомата называются совместимыми, если любое допустимое входное слово автомат перерабатывает в непротиворечивые выходные слова независимо от того, какое из этих двух состояний было выбрано в качестве начального ($s' \sim s''$).

Отношение совместимости обладает следующими свойствами:

- 1) рефлексивность, т.е. $s' \sim s'$;
- 2) симметричность, т.е. если $s' \sim s''$, то из этого следует: $s'' \sim s'$;
- 3) отсутствие транзитивности, т.е. если $s' \sim s_k$ и $s'' \sim s_k$, то из этого не следует, что $s' \sim s''$.

Отношение совместимости разбивает все множество состояний S на классы совместимости: $S = Q_1 \cup Q_2 \cup Q_3 \cup \dots \cup Q_q$.

Класс совместимости $Q_i \in S$ – это подмножество состояний попарно совместимых между собой, причем $Q_i \cap Q_j \neq \emptyset$, если $i \neq j$, т.е. пересечение любых двух классов совместимости не обязательно является пустым множеством и следовательно одно и то же состояние может входить в разные классы совместимости.

В общем случае число классов совместимости больше, чем число классов эквивалентности и даже может и превышать исходных число состояний. Классы совместимости бывают простые и максимальные.

Простым классом совместимости называют подмножество состояний, попарно совместимых между собой.

Максимальным классом совместимости называют класс, который не является подмножеством какого – либо другого класса совместимости.

Два необходимых условия совместимости состояний автомата Мили.

Два состояния s' и s'' являются совместимыми, если:

- 1) при переходе из этих состояний под воздействием любого одинакового входного символа p_k функции выхода, если они определены, должны совпадать, т.е: $\psi'(s', p_k) = \psi''(s'', p_k)$;
- 2) при переходе из этих состояний под воздействием любого одинакового входного символа p_k переход, если он определен, должен осуществляться в одно и то же или в совместимые состояния, т.е. если $s_i = \phi'(s', p_k)$ и $s_j = \phi''(s'', p_k)$, то следует, что $s_i \sim s_j$.

На рис. 2.17 приведен пример совместимых состояний и их объединения. Состояния $s' \sim s''$, так как из s' не определен переход по символу p_1 , а из состояния s'' по символу p_2 .

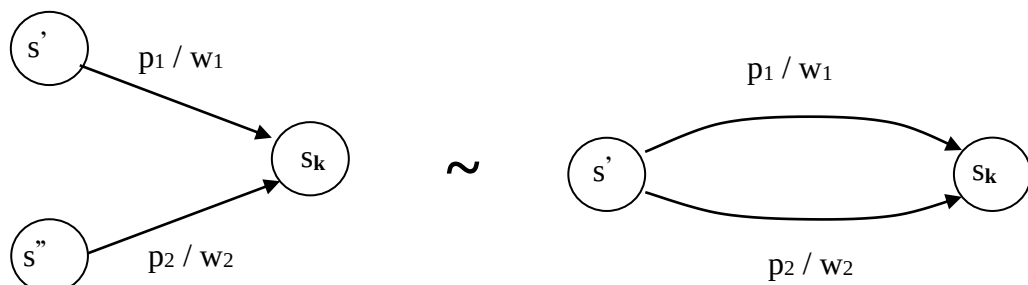


Рис. 2.17

Алгоритм минимизации числа состояний, не полностью определенного (частичного) с помощью треугольной матрицы.

Минимизация числа состояний заключается в уменьшении числа состояний за счет объединения совместимых состояний.

Пусть задан таблицей переходов и выхода не полностью определенный автомат Мили, имеющий k – состояний.

Алгоритм состоит из выполнения следующих шагов:

1. Строится треугольная матрица и аналогично, как в предыдущем алгоритме для полностью определенного автомата, определяются пары совместимых состояний.
2. Формируются максимальные классы совместимости.
3. Выбирается минимально необходимое число максимальных классов совместимости, для которых выполняются условия полноты и замкнутости.
4. Если условия полноты и замкнутости выполняются для выбранного подмножества классов совместимости, то каждому классу совместимости присваивается символ нового состояния, а если не выполняются, то выбранное подмножество классов совместимости корректируется.
5. Переписывается исходная таблица переходов так, чтобы при замене символа старого состояния на символ нового состояния не нарушилось условие замкнутости, далее объединяются строки таблицы, соответствующие совместимым состояниям.

Подмножество классов совместимости называется полным, если каждое состояние исходного автомата входит хотя бы в один из выбранных классов совместимости этого подмножества.

Подмножество классов совместимости называется замкнутым, если для любого его класса подмножество состояний, следующее за этим классом, целиком содержится хотя бы в одном из выбранных классов совместимости этого подмножества.

Проверка условия полноты производится с помощью таблицы покрытий, в которой столбцы соответствуют состояниям исходного автомата, а строки – выбранным классам совместимости.

Проверка условия замкнутости производится после нахождения подмножества состояний, следующих за каждым из выбранных классов совместимости. Для нахождения подмножества состояний $E \in S$, следующее за подмножеством состояний $D \in S$, необходимо определить по исходной таблице переходов подмножество состояний, в которые переходит автомат по одному и тому же входному символу из состояний подмножества D .

Если условие замкнутости не выполняется, то можно сузить какой либо класс совместимости, не нарушая при этом условие полноты, удалить из него состояние, которое нарушает условие замкнутости, или

расширить число выбранных классов совместимости, добавив новый класс совместимости, необходимый для выполнения условия замкнутости. Для нового выбранного класса совместимости необходимо также проверить выполнения условия замкнутости.

Пример минимизации частичного автомата Мили, заданного таблицей переходов и выхода (табл. 2.10).

$P = \{ a, b, c, d, e \}$, $W = \{ 00, 01, 10, 11 \}$, $S = \{ s_1, s_2, s_3, s_4, s_5, s_6, s_7 \}$.

В таблице переходов и выхода (табл. 2.10) для наглядности вместо символов состояний указаны только их индексы.

Таблица 2.10

$s_i \backslash p_k$	a	b	c	d	e
1	1 / 00	5 / *1	---	---	6 / **
2	5 / *0	4 / 0*	---	---	---
3	4 / 11	---	---	---	6 / 00
4	6 / *1	6 / 00	---	2 / 0*	---
5	---	7 / 0*	---	4 / *0	---
6	---	---	3 / *0	---	2 / 10
7	---	2 / **	1 / 00	---	---

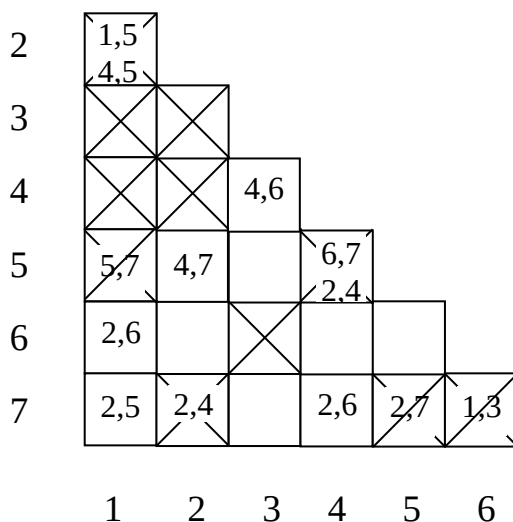


Рис. 2.18

На рис. 2.18 приведена треугольная матрица, заполненная по изложенному выше алгоритму. Клетки, не содержащие «X», соответствуют следующим парам совместимых состояний: 1 ~ 6, 1 ~ 7, 2 ~ 5, 2 ~ 6, 3 ~ 4, 3 ~ 5, 3 ~ 7, 4 ~ 6, 4 ~ 7, 5 ~ 6 и из них можно образовать максимальные классы совместимости например, с помощью дерева.

Для этого строим дерево классов, разбивая на каждом шаге множество состояний на два, одно из которых без состояний, несовместимых с i -тым, другое не содержит само i -тое состояние (рис.2.19):

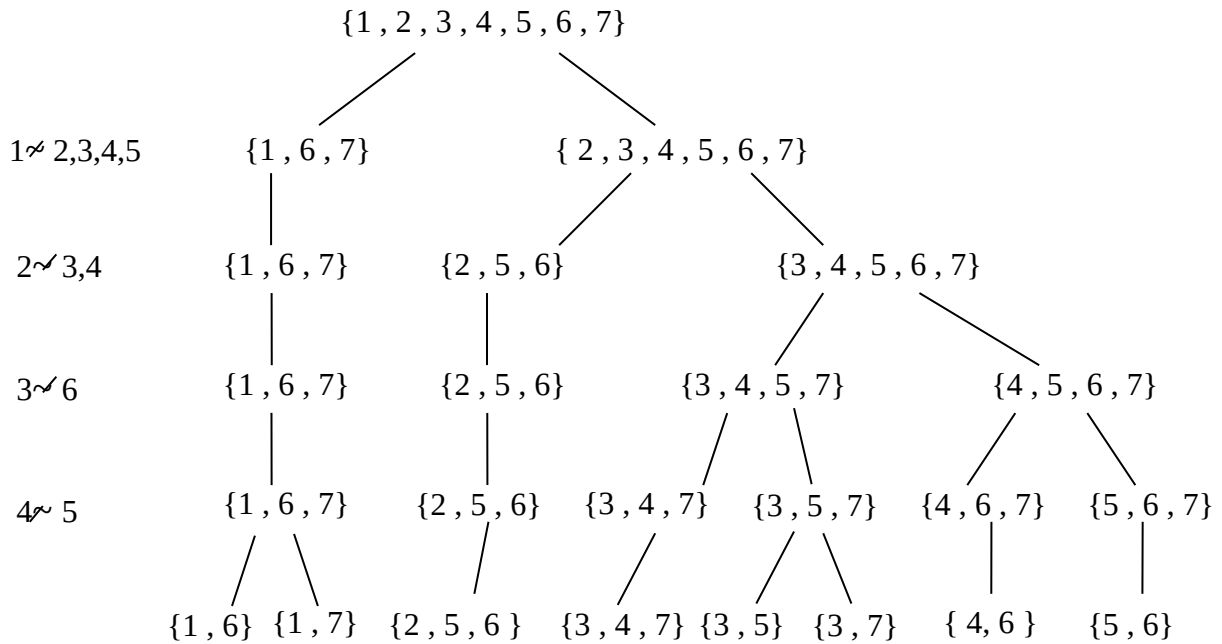


Рис. 2.19

Получаются следующие максимальные классы $Q_1 = \{1, 6\}$, $Q_2 = \{1, 7\}$, $Q_3 = \{2, 5, 6\}$, $Q_4 = \{3, 4, 7\}$, $Q_5 = \{3, 5\}$, $Q_6 = \{4, 6\}$.

Для выполнения условия полноты достаточно выбрать следующее минимальное число максимальных классов совместимости: Q_1, Q_3, Q_4 .

Для проверки условия замкнутости по таблице переходов определяются подмножества состояний, следующих за выбранными классами совместимости (рис. 2.20).

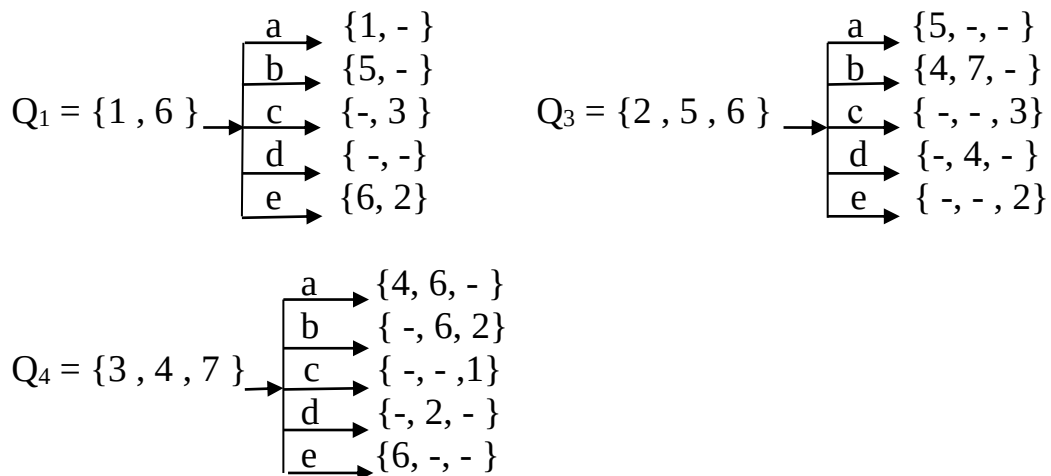


Рис. 2.20

Из рис. 2.20 видно, что подмножество $\{4, 7\}$, следующее по символу «b» за классом Q_3 , входит в выбранный класс совместимости Q_4 – это является соблюдением условия замкнутости.

Подмножество $\{4, 6\}$, следующее по символу «a» за классом Q_4 , не входит ни в один из выбранных классов совместимости – это является нарушением условия замкнутости.

Для выполнения этого условия необходимо добавить класс совместимости $Q_6 = \{4, 6\}$ и проверить для него выполнение условия замкнутости (рис.2.21).

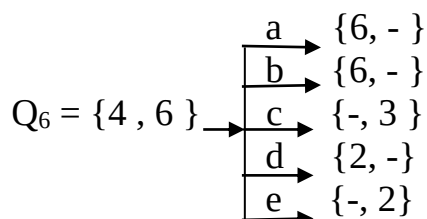


Рис.2.21

Добавление к выбранным классам нового класса Q_6 не нарушает условия замкнутости (рис.2.21). Для упрощения перехода от старых состояний к новым можно сузить класс Q_1 , удалив из него состояние s_6 , не нарушая условий полноты и замкнутости. В результате получается :

$Q_1 = \{ 1 \}$, $Q_3 = \{ 2, 5, 6 \}$, $Q_4 = \{ 3, 4, 7 \}$, $Q_6 = \{ 4, 6 \}$.

Каждому классу сопоставляется символ нового состояния:

$Q_1 - s'_1$, $Q_3 - s'_2$, $Q_4 - s'_3$, $Q_6 - s'_4$. Далее в исходной таблице переходов символ старого состояния заменяется на символ нового состояния, который соответствует классу, в который это старое состояние входит, но с учетом выполнения условия замкнутости. В результате получается таблица переходов и выхода (табл. 2.11).

Таблица 2.11

$s_i \backslash p_k$	a	b	c	d	e
1'	1' / 00	2' / *1	---	---	2' / **
2'	2' / *0	3' / 0*	---	---	---
3'	4' / 11	2' / 00	---	---	2' / 00
4'	4' / *1	4' / 00	---	2' / 0*	---
2'	---	3' / 0*	---	4' / *0	---
4'	---	---	3' / *0	---	2' / 10
3'	---	2' / **	1' / 00	---	---

После совмещения в ней строк, соответствующих совместимым состояниям, получается таблица переходов и выхода минимального

автомата Мили (табл. 2.12) с числом состояний $k^* = 4$ вместо $k = 7$ в исходном автомате.

Таблица 2.12

$s_i \backslash p_k$	a	b	c	d	e
1'	1' / 00	2' / *1	---	---	2' / **
2'	2' / *0	3' / 0*	3' / *0	4' / *0	2' / 10
3'	4' / 11	2' / **	1' / 00	2' / 0*	4' / 00
4'	4' / *1	4' / 00	3' / *0	2' / 0*	2' / 10

Для проверки правильности минимизации промоделируем работу обоих автоматов по переработке входного слова ecadabb.

Таблица 2.13

t_k	t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7
s_i	s_1	s_6	s_3	s_4	s_2	s_5	s_7	s_2
p_j	e	c	a	d	a	b	b	
w_m		**	*0	11	0*	*0	0*	**

Таблица 2.14

t_k	t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7
s'_i	s'_1	s'_2	s'_3	s'_4	s'_2	s'_2	s'_3	s'_2
p'_j	e	c	a	d	a	b	b	
w'_m		**	*0	11	0*	*0	0*	00

Результат работы исходного автомата приведен в табл. 2.13, а результат работы минимального автомата приведен в табл. 2.14. Получены непротиворечивые выходные слова, следовательно, минимизация проведена правильно.

3. СТРУКТУРНЫЙ ЭТАП СИНТЕЗА КОНЕЧНОГО АВТОМАТА

Исходными данными для структурного этапа являются:

- абстрактный автомат, заданный в виде графа или таблицы переходов и выхода;
- система логических элементов, на которых необходимо реализовать автомат;
- тип элемента памяти, который используется в автомате;
- требования к схеме автомата по сложности и быстродействию.

Результатом этапа является функциональная схема автомата из заданных логических элементов.

На структурном этапе автомат рассматривается в виде следующей схемы (рис. 3.1), где:

КС – комбинационная схема, реализующая функции возбуждения элементов памяти и функции выходов структурного автомата;

БП – блок памяти для хранения кода текущего состояния автомата;

ЭП – элемент памяти для хранения одного разряда кода состояния;

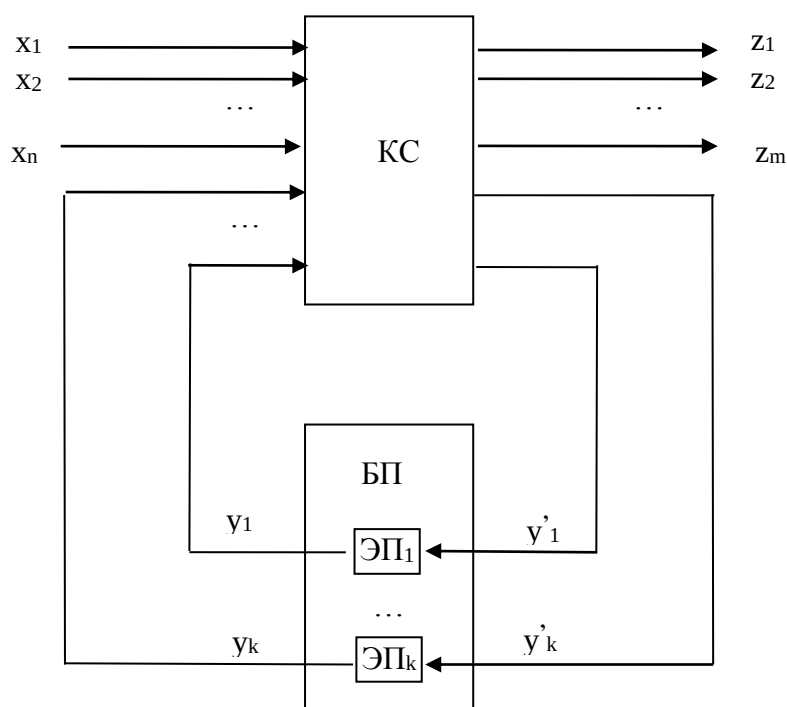


Рис. 3.1

- $x_1, x_2, \dots, x_n$ – входные переменные структурного автомата, которые обозначают разряды кода символов входного алфавита;
- $z_1, z_2, \dots, z_m$ – выходные переменные структурного автомата, которые обозначают разряды кода символов выходного алфавита;
- $y_1, y_2, \dots, y_k$ – внутренние переменные структурного автомата, которые обозначают разряды кода символов состояний;

- $y'_1, y'_2, \dots, y'_k$ – функции возбуждения элементов памяти структурного автомата.

Между объектами абстрактного и структурного автоматов существует следующее однозначное соответствие (табл. 3.1).

Таблица 3.1

Абстрактный автомат	Структурный автомат
$P = \{p_1, p_2, \dots, p_n\}$	$\tilde{X} = \{\alpha_1, \alpha_2, \dots, \alpha_n\}$
$W = \{w_1, w_2, \dots, w_m\}$	$\tilde{Z} = \{\beta_1, \beta_2, \dots, \beta_m\}$
$S = \{s_1, s_2, \dots, s_q\}$	$\tilde{Y} = \{\gamma_1, \gamma_2, \dots, \gamma_q\}$
$s_j = \varphi(s_i, p_k)$	$y'_1 = \varphi'_1(\tilde{Y}, \tilde{X})$ $\dots$ $y'_q = \varphi'_q(\tilde{Y}, \tilde{X})$
функция выхода автомата Мура: $w_j = \psi(s_j)$	$z_1 = \psi'(\tilde{Y})$ $\dots$ $z_m = \psi'_m(\tilde{Y})$
функция выхода автомата Мили: $w_j = \psi(s_j, p_k)$	$z_1 = \psi'_1(\tilde{Y}, \tilde{X})$ $\dots$ $z_m = \psi'_m(\tilde{Y}, \tilde{X})$

$\alpha_i, \beta_j, \delta_k$ – двоичные наборы, являющиеся кодами символов соответствующих алфавитов.

$\tilde{X}, \tilde{Z}, \tilde{Y}$ – множества двоичных наборов, соответствующие алфавитам абстрактного автомата.

$y'_1, \dots, y'_q$ – булевы функции возбуждения элементов памяти, реализующие функции переходов абстрактного автомата.

$z_1, \dots, z_m$ – булевы функции выходов структурного автомата, реализующие функции выхода абстрактного автомата.

Переход от абстрактного автомата к структурному автомату обусловлен двоичной структурой электрических логических элементов, из которых он будет реализован.

Основные этапы структурного синтеза:

- 1) кодирование символов входного и выходного алфавитов;
- 2) кодирование состояний абстрактного автомата;

- 3) после замены символов абстрактного автомата в исходной таблице переходов и выхода на их коды получается кодированная таблица переходов и выходов, которая описывает уже структурный автомат;
- 4) на основе полученной кодированной таблицы строится система булевых функции возбуждения элементов памяти и выходов автомата;
- 5) совместная минимизация полученной системы булевых функций;
- 6) реализация минимальной системы булевых функций из заданных логических элементов;
- 7) построение блока памяти автомата;
- 8) построение функциональной схемы всего автомата.

3.1. ТИПЫ ЭЛЕМЕНТОВ ПАМЯТИ

В качестве элементов памяти структурного автомата используется двоичный элемент – триггер, который может находиться в одном из двух устойчивых состояний «1» или «0».

Элементом памяти будем называть автомат Мура, обладающий полной системой переходов и выходов.

Автомат обладает полной системой переходов, если для любой пары состояний s_i и s_j можно указать сигнал, вызывающий переход из s_i в s_j .

Автомат обладает полной системой выходов, если каждому состоянию автомата можно приписать выходной сигнал, отличный от выходных сигналов других состояний.

Триггер обычно имеет два выхода: прямой и инверсный.

Состояние триггера определяется значением на его прямом выходе.

Основные типы триггеров.

Существует четыре основных логических типов триггеров:

- два одноходовых: D –триггер и T –триггер;
- два двухходовых: RS –триггер и JK-триггер.

D – триггер или триггер – задержка (delay).

На выходе триггера повторяется значение на его входе.

Граф автомат Мура, задающий D – триггер (рис. 3.2). Дуги графа помечены значениями входа D.

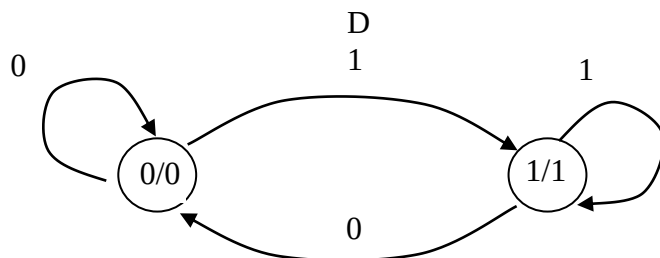


Рис. 3.2

Таблица 3.2

D \ q	0	1
0	0	1
1	0	1

При задании триггера графом автомата Мура обозначение выхода q совпадает с обозначением состояния автомата и имеет то же значение, поэтому таблица переходов триггера не содержит выходной символ (табл. 3.2).

$q(t) \rightarrow q(t+1)$	D
0 $\rightarrow$ 0	0
0 $\rightarrow$ 1	1
1 $\rightarrow$ 0	0
1 $\rightarrow$ 1	1

Рис. 3.3

Другим способом задания работы триггера является матрица переходов (рис. 3.3), где $q(t)$ – состояние триггера в текущем такте; $q(t+1)$ – в следующем при подаче на вход соответствующего значения D.

На рис. 3.4 приведено условное обозначение D – триггера.

T- триггер или счетный триггер (toggle – кувыркатся).

При подаче на вход единицы триггер меняет свое состояние на противоположное.

Граф автомата Мура, задающий T – триггер (рис. 3.5). Дуги графа помечены значениями входа T. Таблица переходов – (табл. 3.3).

Таблица 3.3

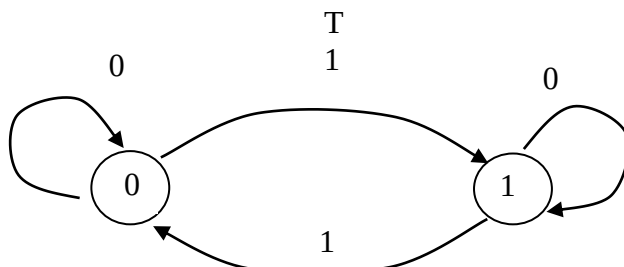


Рис. 3.5

	T	
	0	1
q		
0	0	1
1	1	0

Матрица переходов T – триггера приведена на рис. 3.6, а условное обозначение на рис. 3.7.

$q(t) \rightarrow q(t+1)$	T
0 $\rightarrow$ 0	0
0 $\rightarrow$ 1	1
1 $\rightarrow$ 0	1
1 $\rightarrow$ 1	0

Рис. 3.6

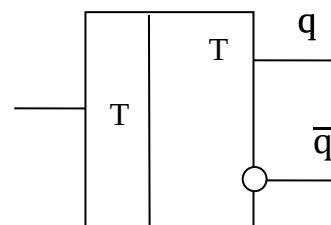


Рис. 3.7

RS – триггер.

S – вход установки «1» (set), т.е. при подаче «1» на вход S, триггер переходит в состояние «1» (установка 1).

R – вход установки «0» (reset), т.е. при подаче «1» на вход R, триггер переходит в состояние «0» (сброс в 0).

Подача на входы S и R двух единиц одновременно запрещена.

Подача на входы S и R двух нулей одновременно не меняет состояние триггера.

Граф автомат Мура, задающий RS – триггер (рис. 3.8). Дуги графа помечены значениями входов S R. Таблица переходов – (табл. 3.4).

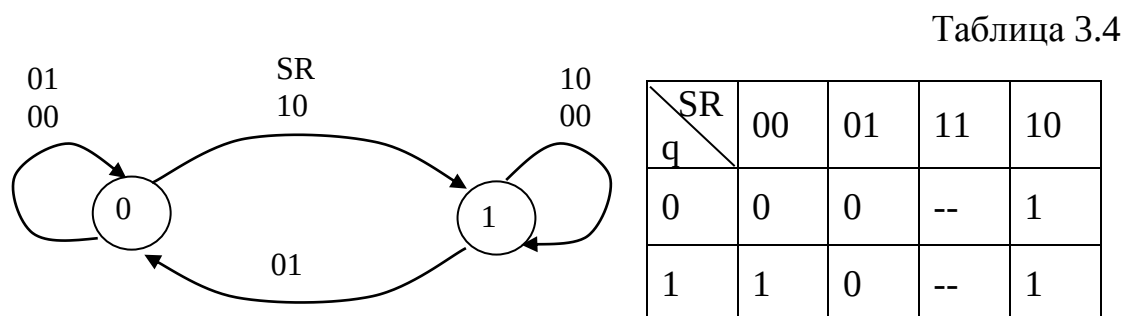


Рис.3.8

Матрица переходов RS – триггера приведена на рис. 3.9, а условное обозначение на рис. 3.10. Символ «*» определяет любое значение.

Q(t) ·q(t+1)	S R
0 → 0	0 *
0 → 1	1 0
1 → 0	0 1
1 → 1	* 0

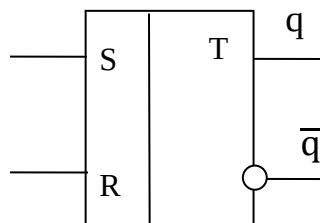


Рис. 3.10

Рис. 3.9

JK триггер.

J – вход установки «1» (jump), т.е. при подаче «1» на вход J, триггер переходит в состояние «1» (установка 1).

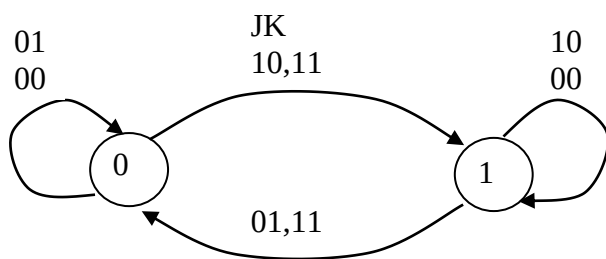
K – вход установки «0» (kill), т.е. при подаче «1» на вход K, триггер переходит в состояние «0» (сброс в 0).

Подача на входы J и K двух единиц одновременно меняет состояние триггера на противоположное.

Подача на входы J и K двух нулей одновременно не меняет состояние триггера.

Граф автомат Мура, задающий JK – триггер (рис. 3.11). Дуги графа помечены значениями входов J K. Таблица переходов – (табл. 3.5).

Таблица 3.5



JK \ q	00	01	11	10
0	0	0	1	1
1	1	0	0	1

Рис. 3.11

Матрица переходов JK – триггера приведена на рис. 3.12, а условное обозначение на рис. 3.13. Символ «*» определяет любое значение.

Q(t) ·q(t+1)	J K
0 → 0	0 *
0 → 1	1 *
1 → 0	* 1
1 → 1	* 0

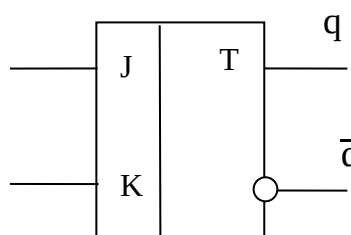


Рис. 3.13

Рис. 3.12

3.2. ПРИМЕР СТРУКТУРНОГО СИНТЕЗА СИНХРОННОГО АВТОМАТА

Автомат называется синхронным, если его состояния меняются в строго определенные моменты времени, задаваемые с помощью специальных сигналов синхронизации.

Пусть задан абстрактный автомат Мили в виде графа (рис. 3.14) или таблицы переходов и выхода (табл. 3.6). Этот автомат был получен ранее в результате выполнения абстрактного этапа канонического метода синтеза.

Таблица 3.6

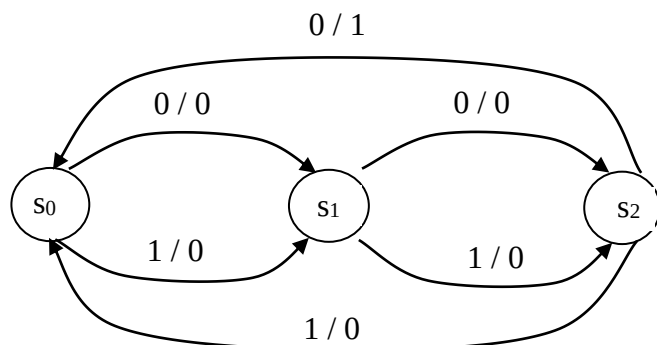


Рис. 3.14

Pj \ Si	0	1
s0	s1/0	s1/0
s1	s2/0	s2/0
s2	s0/1	s0/0

Требуется построить функционально – логическую схему автомата из логических элементов типа «И» (конъюнкторов) и типа «НЕ» (инверторов). В качестве элемента памяти задан RS – триггер.

Логическая схема автомата должна иметь минимальное число логических элементов.

Исходный абстрактный автомат имеет следующие алфавиты: $P = \{0,1\}$, $W = \{0,1\}$ и их кодирование проводить не надо так как они двоичные.

Множество состояний $S = \{s_0, s_1, s_2\}$. Для кодирования трех состояний достаточно два разряда. Произвольно закодируем состояния следующим образом (табл. 3.7) и заменим в исходной таблице переходов и выхода (табл. 3.6) абстрактные символы состояний на соответствующие им коды. Полученная кодированная таблица переходов и выходов (табл. 3.8).

Таблица 3.7

	y_1	y_2
s_0	0	1
s_1	1	1
s_2	1	0

Таблица 3.8

$y_1 y_2 \backslash x$	0	1
0 1	11 / 0	11 / 0
1 1	10 / 0	10 / 0
1 0	01 / 1	01 / 0

Структурная схема для исходного автомата представлена на рис. 3.15.

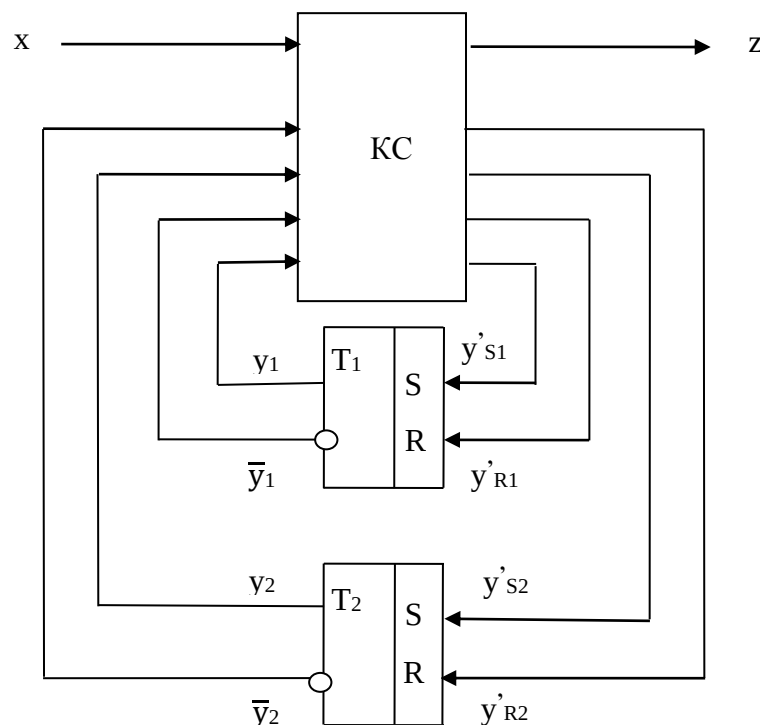


Рис. 3.15

Из структурной схемы автомата на рис. 3.15 видно, что комбинационная схема реализует систему следующих пяти булевых функций: z – функция выхода, y'_{s1} , y'_{r1} – функции возбуждения триггера T_1 , y'_{s2} , y'_{r2} – функции возбуждения триггера T_2 , зависящих от трех переменных: x , y_1 , y_2 . Для определения функции выхода z достаточно кодированной таблицы переходов и выходов (табл. 3.8), а для определения функций возбуждения y'_{s1} , y'_{r1} , y'_{s2} , y'_{r2} необходима еще матрица переходов используемого триггера (рис. 3.9).

Для упрощения нахождения этих булевых функций удобно использовать *таблицу функций возбуждения и выходов* (табл. 3.9), которая строится предварительно на основании кодированной таблицы переходов и выходов (табл. 3.8), и матрицы переходов триггера (в рассматриваемом примере рис. 3.9). Для этого в кодированной таблице переходов и выходов анализируются пары кодов состояний триггеров (до перехода и после перехода) и в соответствующей клетке таблицы функций необходимо вместо кодов состояний записать значения функций возбуждения триггера, которые приведут триггер в требуемое состояние. Значение функции выхода переписывается из кодированной таблицы переходов и выходов. Значения этих функций записываются в клетках таблицы функций следующим образом: $y'_{s1}y'_{r1}, y'_{s2}y'_{r2} / z$.

Например, для реализации перехода автомата из состояния с кодом 01 в состояние с кодом 11 по $x = 0$ необходимо перевести триггер T_1 из 0 в 1 и сохранить единичное состояние триггера T_2 . Для этого, следуя матрице переходов RS – триггера (рис. 3.9), необходимы следующие значения функций возбуждения $y'_{s1} = 1$, $y'_{r1} = 0$, $y'_{s2} = *$, $y'_{r2} = 0$ и если при этом $z = 0$, то в верхнюю, левую клетку табл. 3.9 записывается: 10, *0 / 0.

Таблица 3.9

$x \backslash y_1y_2$	0	1
0 1	10, *0 / 0	10, *0 / 0
1 1	*0, 01 / 0	*0, 01 / 0
1 0	01, 10 / 1	10, 01 / 0

Таблица 3.10

			y_2
			y_1
z			
	—	1	0
	—	0	0
x			

Далее для определения системы булевых функций возбуждения и выходов необходимо построить пять карт Карно для трех переменных и, последовательно просматривая по клеткам полученную таблицу функций возбуждения и выходов (табл. 3.9), одновременно заполнить все пять карт Карно. Например, значения из верхней, левой клетки таблицы функций переписываются в верхнюю, правую клетку соответствующих карт Карно.

Карта Карно для функции выхода – табл. 3.10, а карты Карно для функций возбуждения – табл. 3.11, 3.12, 3.13, 3.14.

Таблица 3.11

y'_{s1} y_2
 y_1

	0	*	
	0	*	

X

Таблица 3.12

y'_{R1} y_2
 y_1

	1	0	0
	1	0	0

X

Таблица 3.13

y'_{s2} y_2
 y_1

	1	0	*
	1	0	*

X

Таблица 3.14

y'_{R2} y_2
 y_1

	0	1	0
	0	1	0

X

После совместной минимизации полученной системы булевых функций с помощью карт Карно получается следующая система булевых функций:

$$z = \bar{x} \bar{y}_2;$$

$$y'_{s1} = \bar{y}_1, \quad y'_{R1} = \bar{y}_2;$$

$$y'_{s2} = \bar{y}_2, \quad y'_{R2} = y_1 y_2.$$

Далее происходит реализация минимальной системы булевых функций из заданных логических элементов, т.е. строится комбинационная схема автомата. К выходам логических схем, реализующих функции возбуждения, подключаются заданные элементы памяти, составляющие блок памяти автомата.

В результате получается следующая функциональная схема автомата (рис. 3.16). Для проверки правильности ее построения необходимо протестировать его работу на конкретном входном слове. На рис. 3.16 показана работа автомата по переработке входного слова $1 \rightarrow 1 \rightarrow 0$. Это слово перерабатывается в выходное слово $0 \rightarrow 0 \rightarrow 1$. Указывается также последовательность значений на выходе каждого логического элемента, причем на выходе триггеров указано на одно значение больше. Это значение соответствует коду начального состояния автомата $s_0 = 01$, в которое он предварительно должен быть установлен (рис. 3.16).

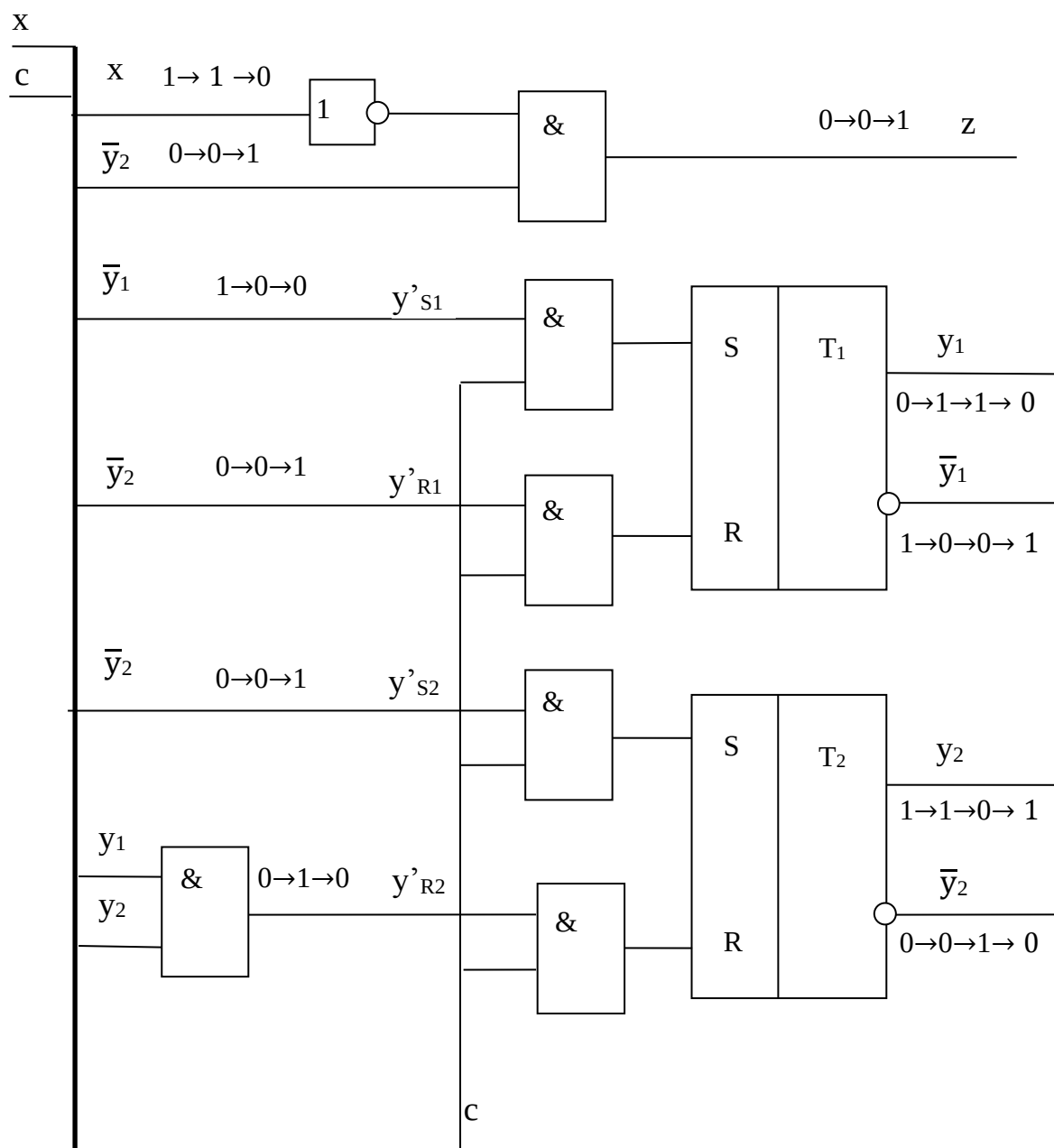


Рис. 3.16

Для обеспечения устойчивой работы автомата в его схеме используется сигнал синхронизации – с. При $c = 0$ значения функций возбуждения непосредственно на входах триггеров равны 0. Это соответствует режиму хранения триггеров и следовательно автомат остается в том же состоянии. При $c = 1$ вычисленные в комбинационной схеме значения функций возбуждения непосредственно поступают на входы триггеров т.е. : $S_1 = y'_{s1}$, $R_1 = y'_{r1}$, $S_2 = y'_{s2}$, $R_2 = y'_{r2}$ и триггеры переключаются и таким образом автомат переходит в новое состояние.

Длительность $c = 1$ должна быть достаточной, чтобы сработал конъюнктор на входе триггера и переключились сами триггеры.

В момент времени, когда переключаются триггеры, значения функций возбуждения не должны меняться, иначе триггер может перейти в незапланированное состояние. В это же время меняются значения на выходе триггеров y_1, y_2 . И они по линии обратной связи поступают на входы комбинационной схемы и могут привести к изменению значений функций возбуждения и выходов. Следовательно, длительность сигнала $c = 1$ должна быть ограничена так, чтобы новые значения y_1, y_2 не успели поступить на входы триггера в этом же автоматном такте.

Пусть: $\Delta\tau$ – время задержки в срабатывании одного логического элемента и если время срабатывания триггера $2\Delta\tau$, то длительность T_1 , когда значение $c = 1$ находится в интервале: $3\Delta\tau < T_1 \leq 3,5\Delta\tau$

Считывать значение функции выхода z можно лишь тогда, когда y_1, y_2 и x не меняются и z имеет установившееся значение после завершения всех переходных процессов в схеме. Это происходит при $c = 1$, т.е. функции возбуждения и выходов необходимо считывать по фронту сигнала синхронизации.

Входной сигнал x необходимо менять по спаду сигнала синхронизации, т.е. при $c = 0$, как только автомат перейдет в новое состояние. Длительность T_0 , когда значение $c = 0$ определяется временем формирования значений функций возбуждения и выходов в комбинационной схеме, определяется числом логических элементов в самой длинной цепочке этой схемы.

Построение временной диаграммы работы схемы автомата

Временная диаграмма отображает работу схемы во времени и строится одновременно с ее логическим моделированием.

Построение временной диаграммы начинается с расчета параметров сигнала синхронизации т.е. определения величин T_1, T_0 и построение сигнала синхронизации на первой оси времени в виде последовательности прямоугольных импульсов, высота которых соответствует логической единице, а ширина – величине T_1 . Расстояние между импульсами соответствует величине T_0 . Импульсы имеют правильную форму, так как формируются генератором синхросигналов.

В момент времени t_0 автомат должен быть установлен в начальное состояние $s_0 = 01$, т.е. $y_1 = 0, y_2 = 1$. Линии, определяющие эти значения, можно продолжить на соответствующих осях до момента времени t_1 , так как при $c = 0$ триггер не может изменить свое состояние.

Далее задается входная последовательность по x : $1 \rightarrow 1 \rightarrow 0$, причем изменение сигнала происходит по спаду сигнала синхронизации в момент времени t_4 .

После нанесения на диаграмму входных воздействий вычисляются по комбинационной схеме для $x = 1, y_1 = 0, y_2 = 1$ значения функций возбуждения и выхода в момент времени t_0 . Эти значения: $y'_{s1} = 1, y'_{r1} = 0, y'_{s2} = 0, y'_{r2} = 0, z = 0$ расставляются на схеме (рис. 3.16).

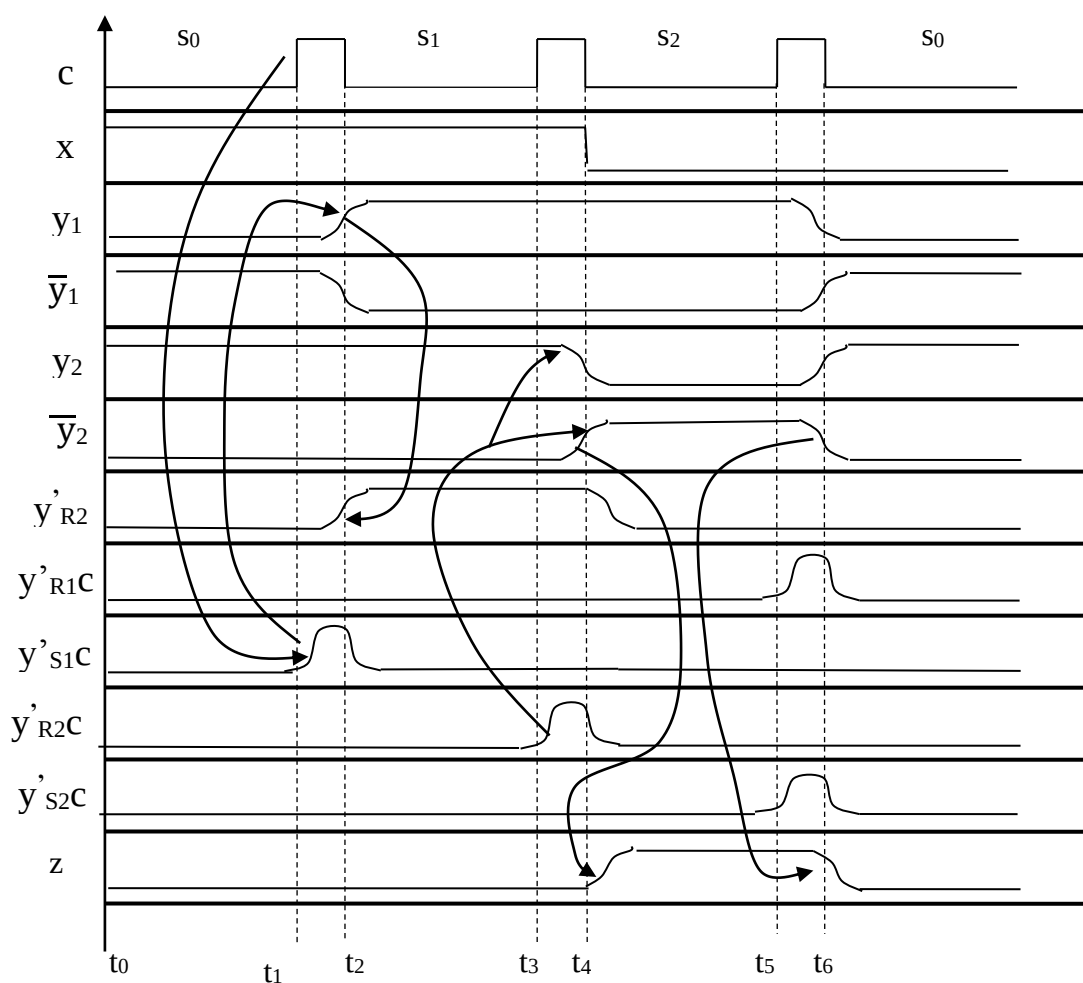


Рис. 3.17

Линии, определенные этими значениями, наносим на соответствующие оси временной диаграммы (рис. 3.17) до момента времени t_1 . Далее по фронту сигнала синхронизации в момент t_1 функции возбуждения поступают на входы триггеров и так как $y'_{s1} = 1$, то первый триггер y_1 установится в 1, т.е. автомат из начального состояния с кодом 01 перейдет в новое состояние с кодом 11. На диаграмме стрелками указывается последовательность изменения значений сигналов. Одновременно со считыванием значений функций возбуждений считывается значение функции выхода $z = 0$.

Эти действия повторяются до момента t_6 , когда перестают поступать сигналы синхронизации $c = 1$ на вход схемы, автомат возвращается в начальное состояние s_0 и прекращает работу по переработке слова.

На временной диаграмме (рис. 3.17) не показаны совпадающие сигналы, так как: $y'_{s1} = \bar{y}_1$, $y'_{r1} = y'_{s2} = \bar{y}_2$.

Недостатком рассмотренной схемной реализации синхронного автомата является то, что достаточно сложно для него построить генератор синхросигналов с такими жесткими требованиями к их длительности.

3.3. КОДИРОВАНИЕ СОСТОЯНИЙ СИНХРОННОГО АВТОМАТА

Основной целью кодирования состояний синхронного автомата является получение минимальной по сложности комбинационной схемы.

Сложность схемы оценивается числом логических элементов или суммарным числом входов ее логических элементов.

Задача кодирования состоит в присвоение каждому состоянию уникального двоичного кода. Число разрядов этого кода q зависит от числа состояний автомата k и находится в интервале:

$$\lceil \log_2 k \rceil \leq q \leq k, \text{ где } \lceil r \rceil - \text{ближайшее целое не меньшее } r.$$

Существуют различные способы кодирования состояний и выбор этого способа существенно влияет на сложность автомата. Способы кодирования различаются по числу используемых элементов памяти (числу разрядов кода).

Способ кодирования с минимальным числом элементов памяти, учитывающий “соседства” состояний на графе автомата.

Два состояния автомата s' и s'' называются “соседями I рода”, если под воздействием хотя бы одного и того же входного символа из них осуществляется переход в одно и то же состояние.

Два состояния автомата s_i и s_j называются “соседями II рода”, если в эти состояния осуществляется переход под воздействием хотя бы одного и того же входного символов из состояний, которые являются “соседями I рода”.

Суть способа кодирования состоит в том, что состояния, являющиеся соседями I рода и II рода, кодируются соседними кодами.

Два двоичных кода одной длины называются соседними, если они различаются значением только в одном разряде.

Фрагмент графа с состояниями, являющимися соседями I рода и II рода представлен, на рис. 3.18.

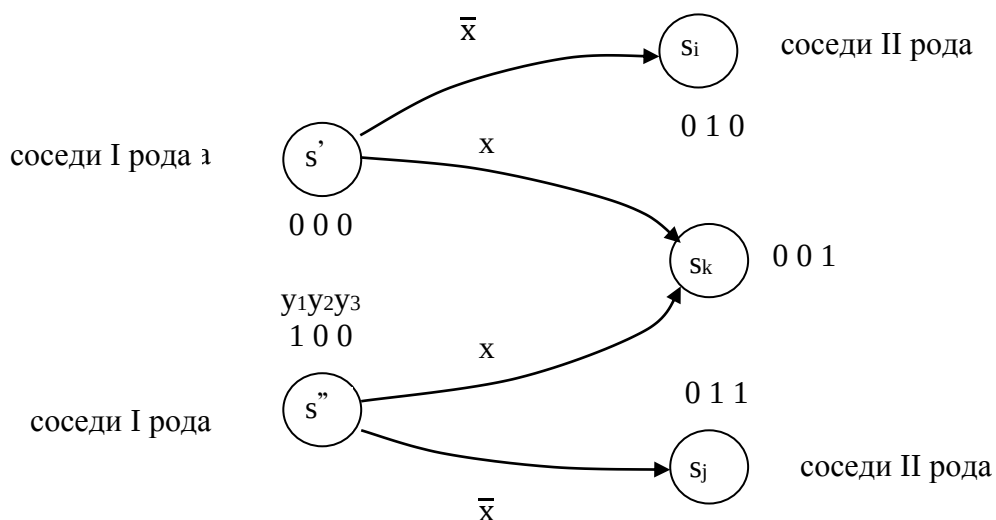


Рис. 3.18

Рассматриваемый способ кодирования является эвристическим, так как он основан на некотором здравом рассуждении, и поэтому приближенным, не гарантирующим оптимальное решение. Это рассуждение можно пояснить следующим примером.

Пусть во фрагменте графа (рис. 3.18) состояния s' и s'' , являющиеся соседями I рода, закодированы следующими соседними кодами: $s' - 000$ и $s'' - 100$, а состояние $s_k - 001$, тогда, если в качестве элемента памяти задан D-триггер, функция возбуждения третьего триггера для перехода из состояний s' и s'' в состояние s_k будет:

$$y'_{D3} = x \bar{y}_1 \bar{y}_2 \bar{y}_3 \vee x y_1 \bar{y}_2 \bar{y}_3 = x \bar{y}_2 \bar{y}_3.$$

Первая конъюнкция определяет переход из состояния s' в состояние s_k , а вторая из состояния s'' в состояние s_k , устанавливая третий триггер в единицу.

Из этого примера видно, что кодирование соседними кодами состояний, являющихся соседями I рода, позволяет получить в функциях возбуждения склеивающиеся между собой конъюнкции и за счет этого упростить комбинационную схему.

В этом же фрагменте состояния s_i и s_j , являющиеся соседями II рода, закодированы следующими соседними кодами: $s_i - 010$ и $s_j - 011$, тогда функции возбуждения второго и третьего триггера:

$$y'_{D2} = \bar{x} \bar{y}_1 \bar{y}_2 \bar{y}_3 \vee \bar{x} y_1 \bar{y}_2 \bar{y}_3 = \bar{x} \bar{y}_2 \bar{y}_3;$$

$$y'_{D3} = x \bar{y}_2 \bar{y}_3 \vee \bar{x} y_1 \bar{y}_2 \bar{y}_3.$$

Первая конъюнкция функции y'_{D2} определяет переход из состояния s' в состояние s_i , а вторая из состояния s'' в состояние s_j , устанавливая второй триггер в единицу. Вторая конъюнкция функции y'_{D3} определяет переход из состояния s'' в состояние s_j , устанавливая третий триггер в единицу.

Таким образом, кодирование соседними кодами состояний, являющихся соседями II рода, позволяет получить в функциях возбуждения склеивающиеся между собой конъюнкции в большем числе функций, чем при другом кодировании.

Обычно не хватает соседних кодов при назначении их соседям I и II рода и поэтому соседние коды надо в первую очередь назначать состояниям, являющихся соседями I рода, а затем назначать соседние коды состояниям, являющихся соседями II рода. Если соседних кодов не хватает уже для соседей I рода, то в первую очередь их надо назначать состояниям, у которых “степень соседства” больше. Под “степенью соседства” понимается количество одинаковых входных символов, по которым осуществляется переход в одно и то же состояние.

Пример кодирования состояний с учетом соседства состояний.

Пусть задана таблица переходов автомата (табл. 3.15).

Определение состояний, являющихся соседями I и II рода, удобно осуществлять по инверсной таблице переходов. Эта таблица отличается от известной (прямой) таблицы переходов тем, что в ее клетках указываются

состояния, из которых осуществляется переход по входному символу p_j в состояние s_i , которым отмечена строка (табл. 3.16).

Таблица 3.15

$s_j \backslash p_i$	0	1
s_0	s_1	s_2
s_1	s_3	s_2
s_2	s_4	s_0
s_3	s_2	s_1
s_4	s_4	s_0

Таблица 3.16

0	1	$p_i \backslash s_j$
---	s_2, s_4	s_0
s_0	s_3	s_1
s_3	s_0, s_1	s_2
s_1	---	s_3
s_2, s_4	---	s_4

В инверсной таблице переходов состояния, являющиеся соседями I рода, расположены в одной клетке: это s_0, s_1 и s_2, s_4 . Состояния, являющиеся соседями II рода, это те состояния, в которые существуют переход из состояний, являющихся соседями I рода, расположенных в одном столбце. В примере это состояния s_1 и s_3 , потому что в s_1 существует переход по 0 из состояния s_0 , а в s_3 существует переход по 0 из состояния s_1 , являющихся соседями I рода.

Для назначения соседних кодов найденным соседям I рода: s_0, s_1 и s_2, s_4 , а также соседям II рода: s_1 и s_3 удобно использовать таблицу, клетки которой закодированы зеркальным кодом Грея, как в карте Карно, но в клетках указываются символы состояний. Состояния, являющиеся соседями I и II рода, записываются в соседних клетках этой таблицы (табл. 3.17).

Таблица 3.17

				y_3
				y_2
	s_2	s_4	s_0	s_1
y_1				s_3

Таблица 3.18

s_i	y_1	y_2	y_3
s_0	0	1	1
s_1	0	0	1
s_2	0	0	0
s_3	1	0	1
s_4	0	1	0

Результат кодирования представлен в кодовой таблице (табл. 3.18).

Способ кодирования, минимизирующий число переключений элементов памяти.

В этом способе кодирования используется минимальное число разрядов кода и он направлен на уменьшение числа переключений триггеров при каждом переходе, так как каждое изменение триггера требует включения конъюнкции в функцию возбуждения этого триггера.

Расстояние между кодами k_i и k_j по Хэммингу – это число разрядов, в которых значения соответствующих разрядов не совпадают. Для вычисления этого расстояния используется операция суммирования по модулю два. Пусть $k_i = 11001$, $k_j = 01100$ тогда:

$$\begin{array}{r} k_i \quad 11001 \\ + k_j \quad 01100 \\ \hline 10101 \end{array}$$

Число единиц в результате сложения по модулю два: $d_{ij} = 3$.

Чтобы определить общее число изменения значений элементов памяти, надо вычислить сумму расстояний по всем переходам.

Алгоритм кодирования:

1. Определяется множество пар состояний M , между которыми существует переход.
2. Произвольно выбирается из M любая пара состояний s_i и s_j , одно из них s_i кодируется всеми нулями, а другое s_j – нулями и одной единицей в самом правом разряде (00...01).
3. Закодированные пары состояний удаляются из множества M и если все пары закодированы, то процесс кодирования завершается.
4. Из множества M произвольно выбирается пара, в которой одно состояние уже закодировано, а другое нет. Незакодированное состояние обозначается s_q .
5. Из множества M выбираются пары, в которые входит состояние s_q . Подмножество таких пар обозначается M_q . Множество уже закодированных состояний из M_q , обозначается V_q .
6. Если $V_q = \emptyset$, то осуществляется переход к пункту 4.
7. Для любого состояния из V_q строится множество кодов C_q^d , находящихся на расстоянии d от кода состояния S_q , где вначале $d = 1$. Если таких кодов нет, то d увеличивается на 1, и так до тех пор, пока не будет найдено множество кодов, находящихся на минимальном расстоянии.
8. Для каждого кода множества C_q^d определяется сумма расстояний по Хеммингу, с каждым кодом из V_q , выбирается код, имеющий минимальную сумму, и этот код приписывается S_q . Далее переход к пункту 3.

Пример кодирования состояний автомата (рис. 3.19):

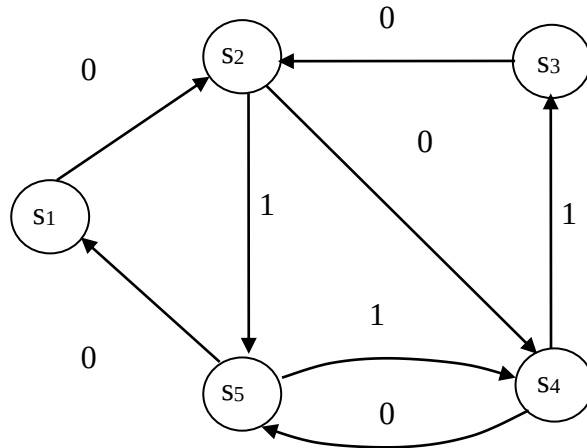


Рис. 3.19

1. Пары состояний $M = \{(1,2), (2,5), (2,4), (3,2), (4,3), (4,5), (5,1)\}$.
 2. Выбираем $\{1,2\} \rightarrow s_1 = 000, s_2 = 001$.
 3. $M = \{(2,4), (2,5), (3,2), (4,3), (4,5), (5,1)\}$.
 4. Выбираем $\{2,4\} \rightarrow s_4, q=4$.
 5. $M_4 = \{(2,4), (4,3), (4,5)\}, B_4 = \{2\}$.
 7. Для кода 001 – $C^1_4 = \{101, 011\}$.
 8. $101 + 001 = 100 \rightarrow d_{101} = 1,$
 $011 + 001 = 010 \rightarrow d_{011} = 1, \rightarrow s_4 = 101$.
 3. $M = \{(2,5), (3,2), (4,3), (4,5), (5,1)\}$.
 4. Выбираем $\{2,5\} \rightarrow s_5, q=5$.
 5. $M_5 = \{(2,5), (4,5), (5,1)\}, B_5 = \{2,4,1\}$.
 7. Для кода 001 – $C^1_5 = \{011\}$.
 Для кода 101 – $C^1_5 = \{(111), (100)\}$.
 Для кода 000 – $C^1_5 = \{(100), (010)\}$.
 8. $011 + 001 = 010, 011 + 101 = 110, 011 + 000 = 110 \rightarrow d_{011} = 5.$
 $111 + 001 = 110, 111 + 101 = 010, 111 + 000 = 111 \rightarrow d_{111} = 6.$
 $100 + 001 = 101, 100 + 101 = 001, 100 + 000 = 100 \rightarrow d_{100} = 4.$
 $010 + 001 = 011, 010 + 101 = 111, 010 + 000 = 010 \rightarrow d_{010} = 6.$
 Минимум $d_{100}=4. \rightarrow s_5 = 100$.
 3. $M = \{(3,2), (4,3)\}$.
 4. Выбираем $\{3,2\} \rightarrow s_3, q=3$.
 5. $M_3 = \{(3,2), (4,3)\}, B_3 = \{2,4\}$.
 Для кода 001 – $C^1_3 = \{011\}$.
 Для кода 101 – $C^1_3 = \{111\}$.
 8. $011 + 001 = 010, 011 + 101 = 110 \rightarrow d_{011} = 3.$
 $111 + 001 = 110, 111 + 101 = 010 \rightarrow d_{111} = 3, \rightarrow s_3 = 011$.
 3. $M = \emptyset$ – конец цикла.
- Результат: $s_1 = 000, s_2 = 001, s_3 = 011, s_4 = 101, s_5 = 100$.

Универсальный способ кодирования состояний синхронного автомата.

При данном способе кодирования используется унитарный код. Унитарным называется код, который содержит только одну единицу, а остальные нули. Это способ кодирования с максимальным числом разрядов. Число разрядов кода равно числу состояний автомата, т.е. состояние s_i кодируется кодом, содержащим единицу в i – том разряде.

Пусть задан фрагмент графа (рис. 3.20), в котором состояние s_1 кодируется кодом 100, а s_2 – кодом 010. Если в качестве элемента памяти задан D-триггер, то для реализации этого перехода функция возбуждения второго триггера: $y'_{D2} = \bar{x} y_1 \bar{y}_2 \bar{y}_3$.

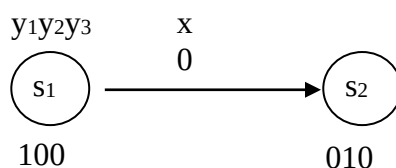


Рис. 3.20

В конъюнкции, обеспечивающей этот переход, переменные y_2 и y_3 являются несущественными, так как признаком того, что автомат находится в состоянии s_1 , является единица в первом разряде, поэтому эту конъюнкцию можно упростить: $y'_{D2} = \bar{x} y_1$. Это является достоинством этого способа кодирования, так как позволяет сразу по графу автомата строить функции возбуждения, которые обычно не требуется минимизировать, но иногда можно упростить.

Пример универсального способа кодирования (рис. 3.21):

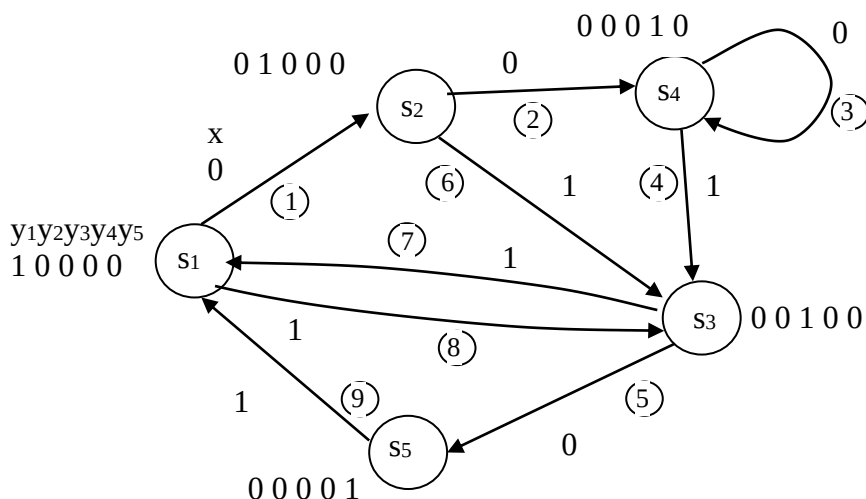


Рис. 3.21

Если в качестве элемента памяти задан D-триггер, то при построении его функций возбуждения необходимо обеспечить его переходы: $0 \rightarrow 1$ и $1 \rightarrow 1$, которые могут быть реализованы поступлением единицы на его вход. Например, для реализации первого перехода (рис. 3.21) из состояния: s_1 в состояние s_2 , необходимо второй триггер установить в единицу, для этого

в функцию возбуждения второго триггера надо включить конъюнкцию, которая имеет значение единицы, когда автомат находится в состоянии s_1 и $x = 0$. Признаком того, что автомат находится в состоянии s_1 , является то, что $y_1 = 1$, поэтому эта конъюнкция имеет вид: $\bar{x} y_1$. Последовательно просматривая все переходы в графе (рис. 3.21), формируются конъюнкции, которые включаются в соответствующие функции возбуждения:

$$y'_{D1} = x y_5 \vee x y_3; \quad \begin{matrix} (9) & (7) \end{matrix}$$

$$y'_{D2} = \bar{x} y_1; \quad (1)$$

$$y'_{D3} = x y_1 \vee x y_2 \vee x y_4; \quad \begin{matrix} (8) & (6) & (4) \end{matrix}$$

$$y'_{D4} = \bar{x} y_2 \vee \bar{x} y_4; \quad \begin{matrix} (2) & (3) \end{matrix}$$

$$y'_{D5} = \bar{x} y_3. \quad (5)$$

Под каждой конъюнкцией указан номер перехода в графе (рис. 3.21), который она обеспечивает.

Если в качестве элемента памяти задан RS-триггер, то при построении его функций возбуждения необходимо обеспечить его переходы: $0 \rightarrow 1$ и $1 \rightarrow 0$. Переход $0 \rightarrow 1$ реализуется поступлением единицы на его вход S. Переход $1 \rightarrow 0$ реализуется поступлением единицы на его вход R. Например, для реализации первого перехода (рис. 3.21) из состояния s_1 в состояние s_2 , необходимо второй триггер установить в единицу, для этого в функцию возбуждения входа S второго триггера надо включить конъюнкцию, которая имеет значение единицы, когда автомат находится в состоянии s_1 и $x = 0$, одновременно необходимо первый триггер сбросить в ноль, для этого в функцию возбуждения входа R первого триггера надо включить ту же самую конъюнкцию. Признаком того, что автомат находится в состоянии s_1 , является то, что $y_1 = 1$, поэтому эта конъюнкция имеет вид $\bar{x} y_1$. Последовательно просматривая все переходы в графе (рис. 3.21), формируются конъюнкции, которые включаются в соответствующие функции возбуждения:

$$y'_{s1} = x y_5 \vee x y_3; \quad \begin{matrix} (9) & (7) \end{matrix}$$

$$y'_{s2} = \bar{x} y_1; \quad (1)$$

$$y'_{s3} = x y_1 \vee x y_2 \vee x y_4; \quad \begin{matrix} (8) & (6) & (4) \end{matrix}$$

$$y'_{s4} = \bar{x} y_2; \quad (2)$$

$$y'_{s5} = \bar{x} y_3; \quad (5)$$

$$y'_{r1} = \bar{x} y_1 \vee x y_1 = y_1; \quad \begin{matrix} (1) & (8) \end{matrix}$$

$$y'_{r2} = \bar{x} y_2 \vee x y_2 = y_2; \quad \begin{matrix} (2) & (6) \end{matrix}$$

$$y'_{r3} = x y_3 \vee \bar{x} y_3; \quad \begin{matrix} (7) & (5) \end{matrix}$$

$$y'_{r4} = x y_4; \quad (4)$$

$$y'_{r5} = x y_5. \quad (9)$$

3.4. СТРУКТУРА АВТОМАТА С ДЕШИФРАТОРОМ

Достоинством способов кодирования с минимальным числом элементов памяти является минимум оборудования для реализации блока памяти, а недостатком – необходимость минимизации системы булевых функций, определяющей функционирование комбинационной схемы.

Достоинством универсального способа кодирования является возможность построить эту систему сразу по графу автомата без её минимизации. Недостатком – максимум оборудования для реализации блока памяти.

Структура с дешифратором (рис. 3.22) позволяет совместить достоинства обоих рассмотренных ранее способов кодирования, т.е. при минимальном числе элементов памяти строить системы булевых функций возбуждения и выходов по графу автомата без дальнейшей минимизации, но с дополнительными затратами оборудования на реализацию дешифратора – DC. В схеме на рис. 3.22 : q – число состояний автомата (число выходов дешифратора), k – минимальное число разрядов кода состояний (число входов дешифратора). На входе дешифратора двоичный код (k разрядов), а на выходе унитарный код (q разрядов), т.е. $q = 2^k$.

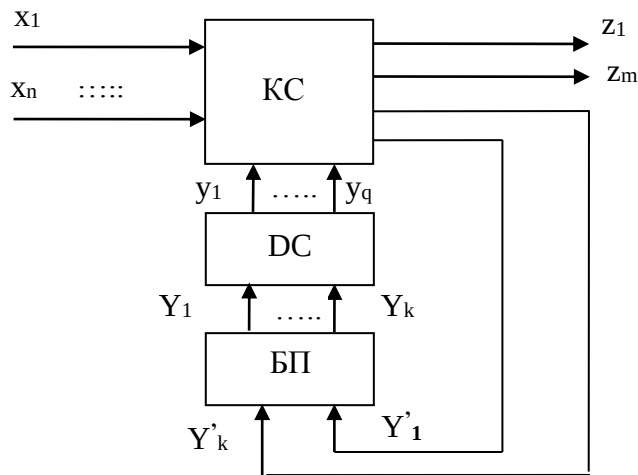


Рис. 3.22

Для построения функций возбуждения элементов памяти необходимо:

каждому состоянию сопоставить два кода, первый с минимальным числом разрядов, а второй с максимальным числом разрядов. Соответствие между минимальным и максимальным кодом для каждого состояния будет задавать функцию дешифратора;

построить функции возбуждения как в способе кодирования с минимальной длиной кода состояний, а конъюнкции, входящие в эти функции, формировать как в способе кодирования с максимальной длиной кода состояний.

Пример построения функций возбуждения для автомата с дешифратором, заданного графом переходов (рис. 3.23).

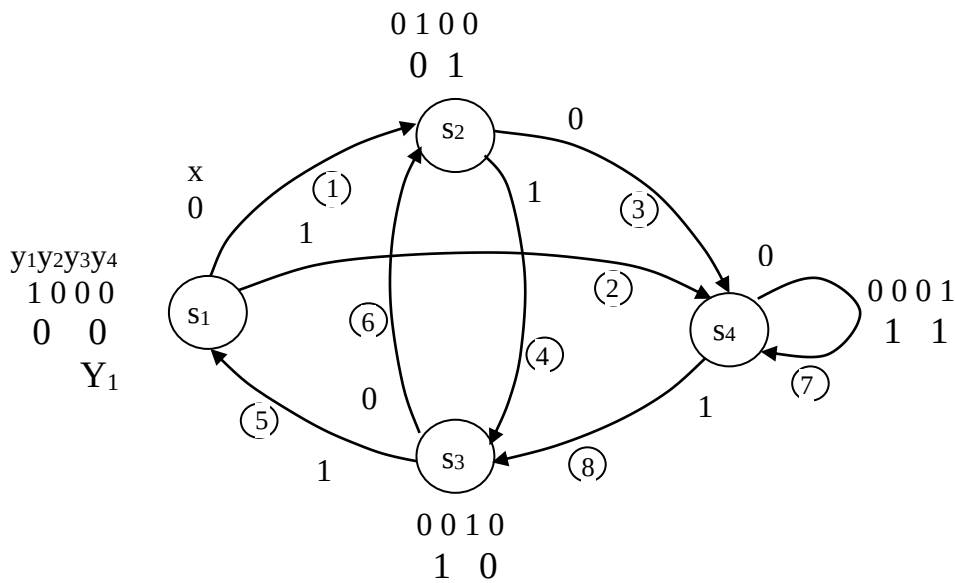


Рис. 3.23

Результат кодирования состояний приведен в табл. 3.19.

Таблица 3.19

s_i	Y_1	Y_2	y_1	y_2	y_3	y_4
s_1	0	0	1	0	0	0
s_2	0	1	0	1	0	0
s_3	1	0	0	0	1	0
s_4	1	1	0	0	0	1

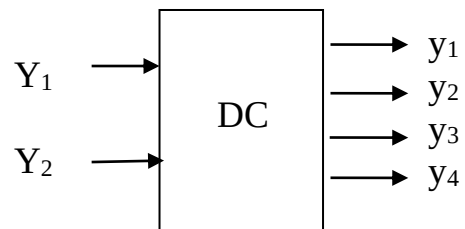


Рис. 3.24

Рассматривая таблицу кодирования как таблицу истинности системы булевых функций, получаем дешифратор с двумя входами и четырьмя выходами (рис. 3.24), так как в блоке памяти хранится двухразрядный код состояния, а на входы комбинационной схемы поступает четырех – разрядный код состояния.

Если в качестве элемента памяти задан D-триггер, то при построении его функций возбуждения необходимо обеспечить его переходы: $0 \rightarrow 1$ и $1 \rightarrow 1$. Например, для реализации перехода (рис. 3.23) из состояния: s_1 в состояние s_2 , необходимо второй триггер Y_2 установить в единицу, для этого в функцию возбуждения второго триггера надо включить конъюнкцию, которая имеет значение единицы, когда автомат находится в состоянии s_1 и $x = 0$. Признаком того, что автомат находится в состоянии

s_1 , является то, что $y_1 = 1$, поэтому эта конъюнкция имеет вид $\bar{x} y_1$. Последовательно просматривая все переходы в графе (рис. 3.23), формируются конъюнкции, которые включаются в соответствующие функции возбуждения:

$$y'_{D1} = x \underset{(2)}{y_1} \vee \bar{x} \underset{(3)}{y_2} \vee x \underset{(4)}{y_2} \vee \bar{x} \underset{(7)}{y_4} \vee x \underset{(8)}{y_4}$$

$$y'_{D2} = \bar{x} \underset{(1)}{y_1} \vee x \underset{(2)}{y_1} \vee \bar{x} \underset{(3)}{y_2} \vee \bar{x} \underset{(6)}{y_3} \vee \bar{x} \underset{(7)}{y_4}$$

Если в качестве элемента памяти задан RS-триггер, то при построении его функций возбуждения необходимо обеспечить его переходы: $0 \rightarrow 1$ и $1 \rightarrow 0$. Например, для реализации второго перехода (рис. 3.23) из состояния s_1 в состояние s_4 , необходимо оба триггера установить в единицу, для этого в функции возбуждения входа S каждого триггера надо включить конъюнкцию, которая имеет значение единицы, когда автомат находится в состоянии s_1 и $x = 1$. Признаком того, что автомат находится в состоянии s_1 , является то, что $y_1 = 1$, поэтому эта конъюнкция имеет вид $x y_1$.

Последовательно просматривая все переходы в графе (рис. 3.23), формируются конъюнкции, которые включаются в соответствующие функции возбуждения:

$$y'_{S1} = x \underset{(2)}{y_1} \vee \bar{x} \underset{(3)}{y_2} \vee x \underset{(4)}{y_2}$$

$$y'_{S2} = \bar{x} \underset{(1)}{y_1} \vee x \underset{(2)}{y_1} \vee x \underset{(6)}{y_3}$$

$$y'_{R1} = x \underset{(5)}{y_3} \vee \bar{x} y$$

$$y'_{R2} = x \underset{(4)}{y_2} \vee x \underset{(8)}{y_4}$$

3.5. ПОСТРОЕНИЕ КОМБИНАЦИОННОЙ СХЕМЫ АВТОМАТА НА ПЗУ и ПЛИМ

Для реализации комбинационной схемы автомата наряду с логическими элементами малой степени интеграции могут быть использованы постоянно-запоминающие устройства (ПЗУ, ROM) и программируемые логические матрицы (ПЛИМ, PLA), относящиеся к схемам средней степени интеграции.

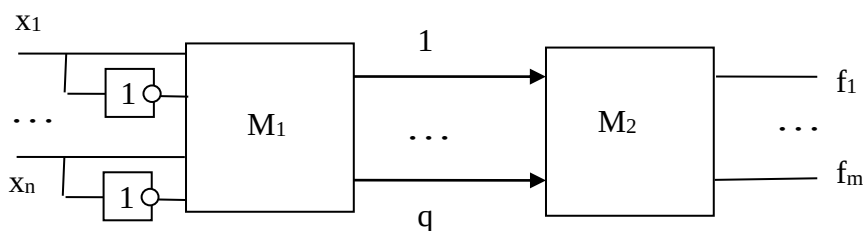


Рис. 3.25

ПЛМ и ПЗУ это элементы, которые предназначены для реализации системы из m булевых функций, зависящих от n переменных.

Оба элемента можно рассматривать как устройство, состоящее из двух матриц M_1 и M_2 (рис. 3.25):

Матрица M_1 предназначена для реализации q различных конъюнкций. Матрица M_2 предназначена для реализации дизъюнкции этих конъюнкций. На вход матрицы M_1 поступают прямые и инверсные значения входных переменных, на выходе получают конъюнкции от n этих переменных. Выходы матрицы M_1 называются промежуточными шинами и являются входами матрицы M_2 . Выходы матрицы M_2 реализуют булевы функции системы.

В ПЗУ матрица M_1 представляет собой дешифратор, т.е. $q = 2^n$ и входной набор значений двоичных переменных $x_1, x_2, \dots, x_n$ рассматривается как адрес ячейки памяти, в которой хранятся значения булевых функций системы $f_1, f_2, \dots, f_m$.

Исходным заданием системы булевых функций для реализации на ПЗУ является таблица истинности системы булевых функций или представление этой системы булевых функций в виде карт Карно до её минимизации. Если булевы функции исходной системы являются не полностью определенными, то для построения таблицы “прошивки” (программирования) ПЗУ необходимо доопределить систему функций на тех наборах, на которых определена хотя бы одна функция системы. Например, для системы функций, представленной ранее картами Карно (табл. 3.10 – 3.14), данные для “прошивки” (программирования) для ПЗУ приведены в табл. 3.20.

ПЛМ, в отличие от ПЗУ, позволяет реализовать систему из m булевых функций от n переменных, содержащую не более q неповторяющихся конъюнкций. В матрице M_1 может быть реализовано q конъюнкций от n входных переменных, в матрице M_2 – m дизъюнкций от конъюнкций, полученных в матрице M_1 .

Таблица 3.20

x	y ₁	y ₂	y's ₁	y'r ₁	y's ₂	y'r ₁	z
0	0	0	1	1	1	0	1
0	0	1	1	0	0	0	0
0	1	0	0	1	1	0	1
0	1	1	0	0	0	1	0
1	0	0	1	1	1	0	0
1	0	1	1	0	0	0	0
1	1	0	0	1	1	0	0
1	1	1	0	0	0	1	0

При реализации системы булевых функций на ПЛМ ее удобно представить в обобщенной табличной форме. Эта табличная форма может, быть непосредственно использована для “прошивки” (программирования) ПЛМ. В таблице “прошивки” должно быть не более q строк, ее левая часть используется для настройки матрицы M_1 , правая – для настройки матрицы M_2 . Например, для системы функций, полученной в результате минимизации с помощью карт Карно (табл. 3.10 – 3.14), её обобщенная табличная форма, приведенная в табл.3.21 является информацией для “прошивки” (программирования) ПЛМ.

$$Z = \bar{x} \bar{y}_2;$$

$$y'_{s1} = \bar{y}_1, \quad y'_{r1} = \bar{y}_2;$$

$$y'_{s2} = \bar{y}_2, \quad y'_{r2} = y_1 y_2.$$

Таблица 3.21

x	y ₁	y ₂	y' _{s1}	y' _{r1}	y' _{s2}	y' _{r2}	z
0	--	0					1
--	0	--	1				
--	--	0		1	1		
--	1	1				1	

Располагая ПЗУ с s адресными входами и t выходами, систему функций с n входными переменными и m функциями в случае $n \leq s$ и $m > t$, т.е. если число функций больше числа выходов ПЗУ, то можно для реализации этой системы использовать несколько ПЗУ по схеме на рис.3.26. Каждое ПЗУ в этой схеме реализует «свою» группу функций.

Аналогично, располагая ПЛМ с s входами, t выходами и q промежуточными шинами, систему m булевых функций от n аргументов с r неповторяющимися конъюнкциями для случая $n \leq s$, $m > t$, $r \leq q$ можно реализовать по схеме на рис.3.26. Все ПЛМ в этой схеме формируют одни и те же конъюнкции, однако каждая реализует свою группу функций.

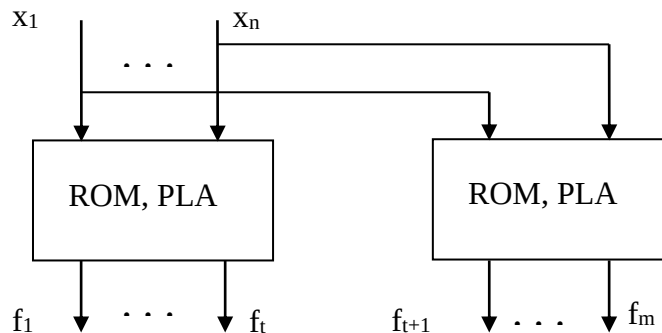


Рис. 3.26

Для случая $n \leq s$, $m \leq t$, $r > q$, т.е. когда существенным является ограничение по числу промежуточных шин, схема приобретает вид на рис. 3.27. Каждая из ПЛМ в этой схеме реализует свою дизъюнкцию группы конъюнкций от общих входных переменных. Электрическим соединением выходов ПЛМ реализуется так называемое “электрическое ИЛИ” каждой такой дизъюнкции, которое допустимо для ПЛМ и недопустимо для логических элементов.

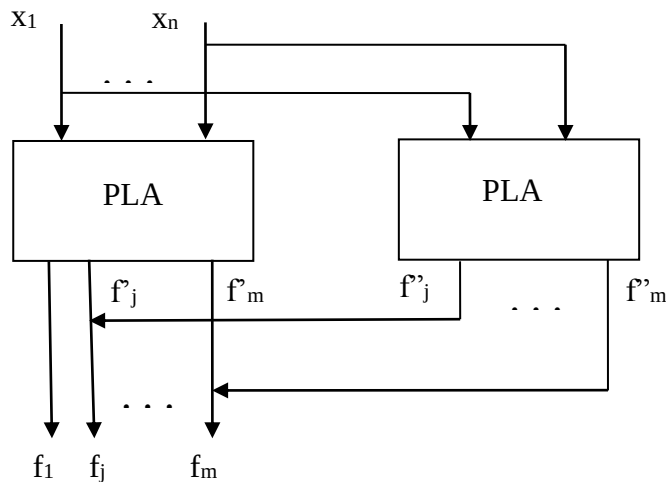


Рис. 3.27

Если число переменных, реализуемой системы булевых функций, превышает число входов ПЛМ или ПЗУ, то необходимо исходную систему декомпозировать на подсистемы, каждая из которых зависит от меньшего, чем исходная, числа переменных.

Условное графическое обозначение ПЗУ и ПЛМ в функциональных схемах приведено на рис. 3.28:

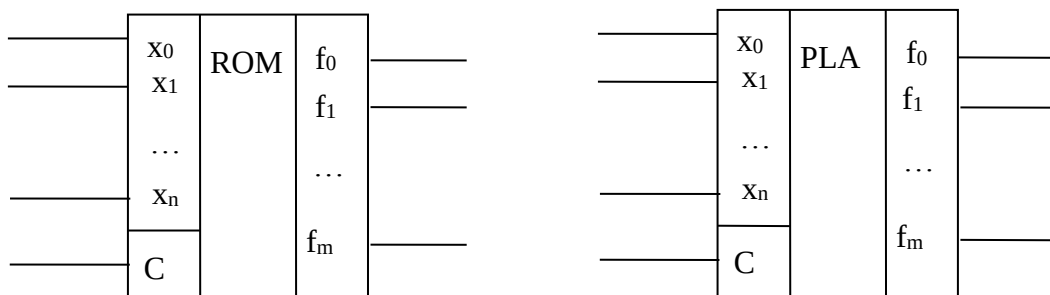


Рис. 3.28

4. СИНТЕЗ МИКРОПРОГРАММНОГО АВТОМАТА

Построение микропрограммного автомата является примером неканонического метода синтеза конечного автомата.

Микропрограммным называется автомат, реализующий микропрограмму. Он является тем управляющим автоматом, который, взаимодействуя с операционным автоматом и обмениваясь с ним двоичными кодами, вместе составляют операционное устройство. Вследствие такого использования микропрограммный автомат можно рассматривать как абстрактный автомат с уже закодированными входными и выходными алфавитами, но еще с абстрактными состояниями.

4.1. ПОСТРОЕНИЕ ГРАФА АВТОМАТА МУРА ПО ГСА

Алгоритм перехода от графической схемой алгоритма (ГСА) микропрограммы к графу автомата Мура.

1. Начальная и конечная вершины ГСА помечаются символом начального состояния s_0 .
2. Все операторные вершины ГСА помечаются произвольно уникальными символами состояний s_i , $1 \leq i \leq n$, независимо от микроопераций, содержащихся в этих вершинах.
3. Строится $n+1$ вершина графа и каждая вершина помечается символом состояния и микрооперацией, содержащейся в соответствующей операторной вершине или ее кодом.
4. Вершина графа s_i соединяется с s_j дугой, отметив эту дугу набором значений логических условий, который допускает переход из операторной вершины, помеченной символом s_i в операторную вершину s_j .

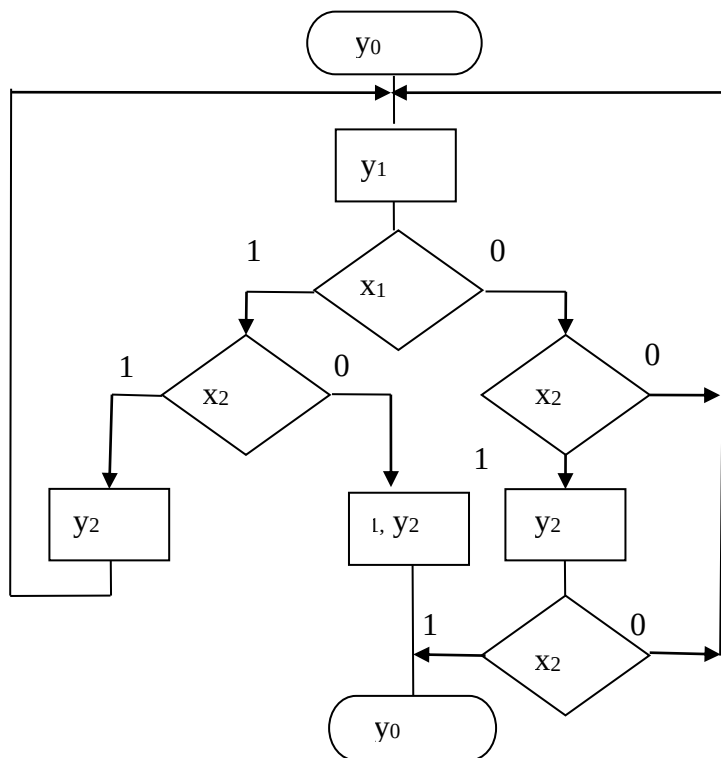


Рис. 4.1

Таблица 4.1

y_i	x_1	x_2
y_0		0
y_1	$\neg$	z
y_2		1
U_0	1	1

Пример построения графа автомата Мура по ГСА микропрограммы (рис. 4.1).

Пусть кроме микропрограммы задана таблица, определяющая влияние микроопераций на значение логических условий (табл. 4.1).

В соответствии с приведенным алгоритмом начальной и конечной вершинам ГСА сопоставляется начальное состояние автомата, отмеченное символом s_0 и микрооперацией y_0 . Остальным операторным вершинам, независимо от содержащихся в них микроопераций, произвольно сопоставляются неповторяющиеся символы состояний s_1, s_2, s_3, s_4 . Необходимо обратить внимание на то, что операторные вершины, отмеченные символами s_2 и s_4 , содержат одну и ту же микрооперацию y_2 , но им сопоставлены два различных состояния.

Граф полностью определенного автомата Мура, реализующего исходную ГСА, представлен на рис.4.2.

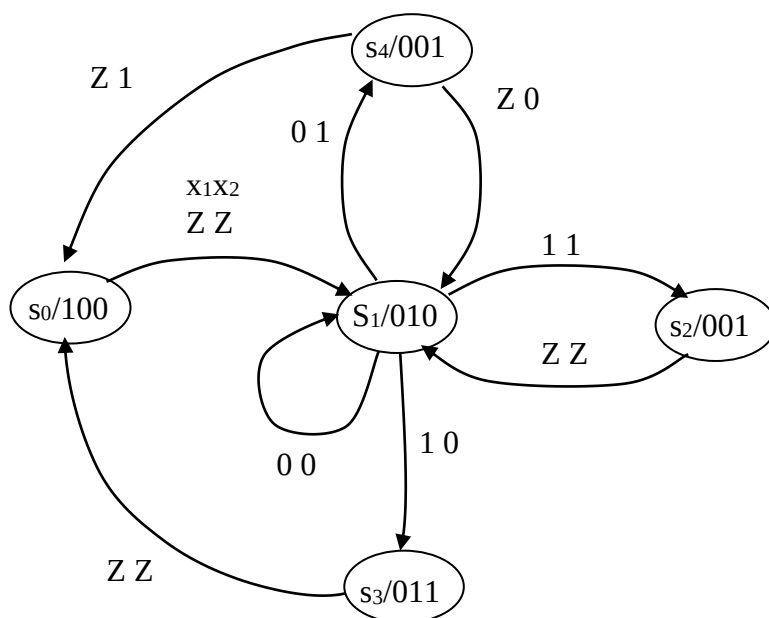


Рис.4.2

Дуги этого графа отмечены троичными кодами наборов значений логических условий, при которых осуществляется соответствующий переход. Если переход возможен независимо от значения логического условия x_i , то в троичном коде этому логическому условию сопоставляется символ z . Если переход происходит независимо от значений всех логических условий, то он отмечается троичным кодом $ZZ...Z$ и число разрядов этого кода равно числу логических условий. Этому коду соответствуют все 2^n наборов значений логических условий и по каждому из 2^n наборов должен существовать переход из каждого состояния. Вершины графа отмечены символом состояния и кодом, разряды которого соответствуют микрооперациям: $y_0\ y_1\ y_2$.

Соответствующая графу (рис.4.2) таблица переходов и выходов построенного автомата Мура представлена в табл.4.2.

Таблица 4.2

$y_0 y_1 y_2$			$x_1 x_2$	0 0	0 1	1 1	1 0
			s_i				
1	0	0	s_0	s_1	s_1	s_1	s_1
0	1	0	s_1	s_1	s_4	s_2	s_3
0	0	1	s_2	s_1	s_1	s_1	s_1
0	1	1	s_3	s_0	s_0	s_0	s_0
0	0	1	s_4	s_1	s_0	s_0	s_1

Использование микропрограммного автомата в цифровых устройствах определяет его синхронную реализацию, поэтому целесообразно перейти к модели автомат Мили. Для этого код входного символа из каждой вершины переносится на дуги входящие, в эту вершину (рис.4.3).

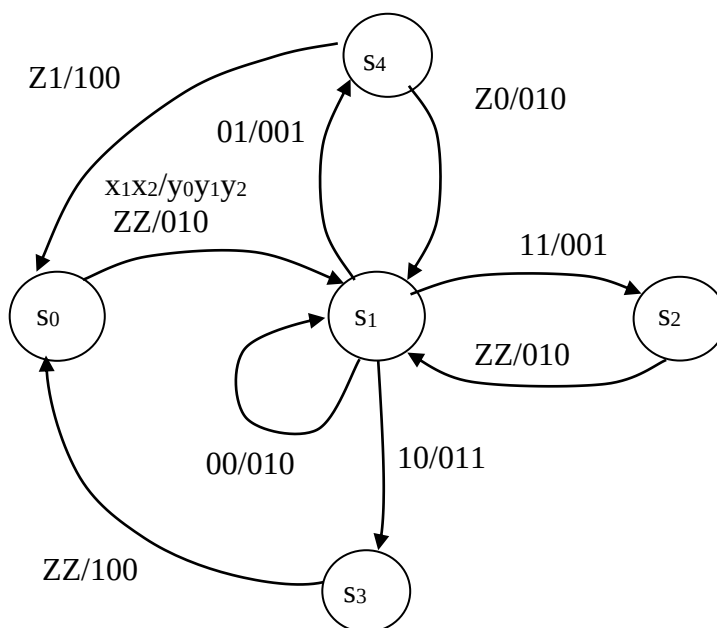


Рис.4.3

В полученном автомате Мили состояния s_0 и s_2 эквивалентны, т.е. $s_0 \equiv s_2$. Получившееся в результате объединения этих эквивалентных состояний новое состояние обозначается: s_0 .

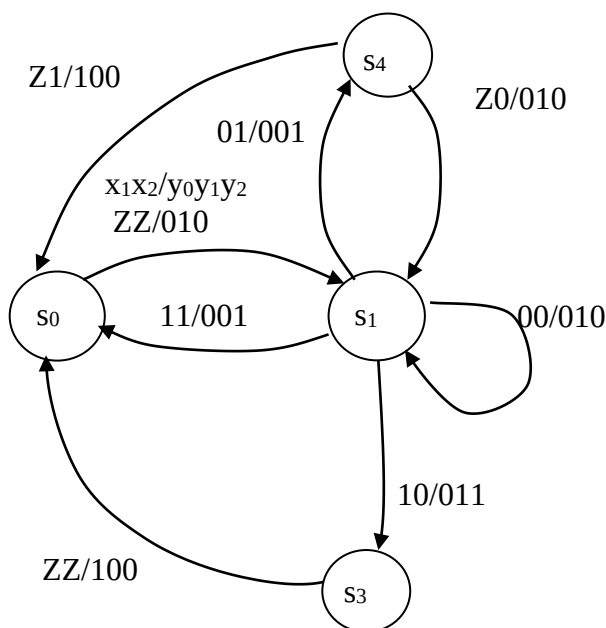


Рис.4.4

В результате получается минимальный автомат Мили (рис.4.4).

4.2. ПОСТРОЕНИЕ НЕ ПОЛНОСТЬЮ ОПРЕДЕЛЕННОГО АВТОМАТА

Учет взаимодействия операционного и управляющего автоматов позволяет получить не полностью определенный (частичный автомат), что дает больше возможностей для упрощения схемы автомата.

При построении графа автомата по ГСА микропрограммы получается полностью определенный автомат, т.е. из каждого состояния существуют переходы по каждому набору значений логических условий. Из рассмотрения работы реальных операционных устройств (например, для сложения двух целых двоичных чисел) можно сделать вывод, что некоторые наборы значений логических условий никогда не поступят на входы управляющего автомата и следовательно, переходы по этим наборам можно исключить из задания автомата. Из учета взаимодействия операционного и управляющего автоматов нужно определить для каждого состояния s_i множество наборов значений логических условий U_i , которые могут поступить на входы управляющего автомата, когда он находится в состоянии s_i .

Для этого необходимо знать:

- множество наборов U_0 , которые могут поступить на входы управляющего автомата, когда он находится в состоянии s_0 ;
- каким образом влияет каждая микрооперация y_j на значение логического условия x_i в результате выполнения этой микрооперации в операционном автомате.

Для определения всех множеств U_i необходимо просмотреть все переходы автомата в виде дерева его переходов, одновременно определяя для каждого состояния s_i множество наборов значений логических условий U_i . Просмотр переходов автомата необходимо продолжать до тех пор, пока все U_i не станут устойчивыми множествами, т.е. пока в них не перестанут появляться новые наборы в множествах U_i .

После получения всех множеств U_i в исходной таблице переходов и выходов полностью определенного автомата надо исключить переходы из каждого состояния s_i по наборам, не содержащимся в его U_i .

Пример получения частичного автомата Мили.

Ранее по исходной ГСА был построен граф минимального полностью определенного автомата Мили (рис.4.4). Таблица переходов и выходов этого автомата приведена в табл. 4.3.

Таблица 4.3

x_1x_2 s_i	00	01	11	10
s_0	$s_1 / 010$	$s_1 / 010$	$s_1 / 010$	$s_1 / 010$
s_1	$s_1 / 010$	$s_4 / 001$	$s_0 / 001$	$s_3 / 011$
s_3	$s_0 / 100$	$s_0 / 100$	$s_0 / 100$	$s_0 / 100$
s_4	$s_1 / 010$	$s_0 / 100$	$s_0 / 100$	$s_1 / 010$

Из исходного задания (табл. 4.1) известно, что перед началом работы автомата на его входы подается набор значений логических условий $x_1 x_2 = 11$, т.е. $U_0 = \{11\}$. Из табл. 4.1 следует, что после выполнения микроопераций значения логических условий меняются следующим образом:

$y_0: x_1 := x_1; \quad x_2 := 0;$

$y_1: x_1 := \bar{x}_1; \quad x_2 := Z$, т.е. значение x_2 может быть любым (0 или 1);

$y_2: x_1 := x_1; \quad x_2 := 1.$

Просмотр переходов для определения множеств U_i начинается из состояния s_0 по входному набору 11. В результате перехода в s_1 с выдачей y_1 определяются первые элементы множества $U_1 = \{00, 01\}$, так как $x_1 := \bar{x}_1$, а $x_2 := z$. По набору 00 автомат останется в s_1 с повторной выдачей y_1 . В результате к U_1 добавятся еще два набора, т.е. 11 и 10. $U_1 = \{00, 01, 11, 10\}$. Далее из s_1 , возможны три различных пути.

Первый путь из s_1 соответствует набору 10, происходит переход из s_1 в s_3 с выдачей $y_1 y_2$. В результате влияния y_1 на x_1 эта переменная изменит свое значение на противоположное. На x_2 влияют обе микрооперации, поэтому ее значение может быть любым. Получается $U_3 = \{01, 00\}$. Далее переход из s_3 в s_0 с выдачей y_0 добавит к U_0 набор 00, т.е. $U_0 = \{11, 00\}$. Второй путь из s_1 по набору 01: $s_1 \rightarrow s_4 \rightarrow s_0$ определяет $U_4 = \{01\}$ и не

добавляет новых наборов к U_0 . Третий путь из s_1 по набору 11 в s_0 с выдачей y_2 также не приводит к появлению новых входных наборов. Окончательно сформированные множества входных наборов компактно представляются следующими троичными векторами:

$$U_0 = \{11, 00\}, U_1 = \{ZZ\}, U_3 = \{0Z\}, U_4 = \{01\}.$$

Схема просмотренных путей в виде дерева переходов приведена на рис. 4.5.

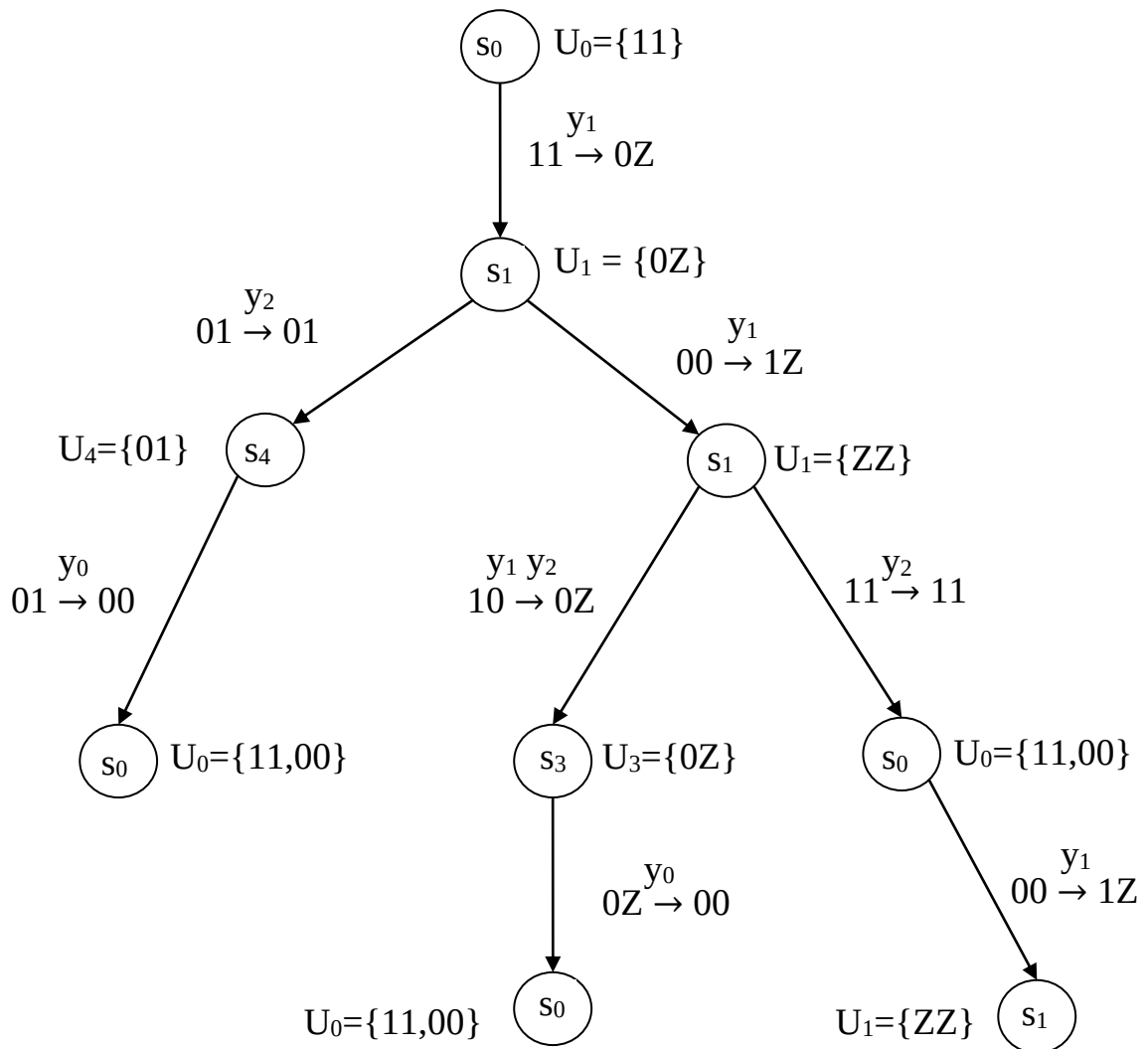


Рис. 4.5

После удаления из табл. 4.3 переходов по наборам, не содержащихся в найденных множествах U_i , получается таблица переходов и выходов частичного автомата (табл. 4.4).

В полученном не полностью определенном автомате состояние s_3 совместимо с состоянием s_4 . После объединения этих состояний и присвоения этому объединенному состоянию символ s_2 получается минимальный частичный автомат Мили (табл. 4.5).

Таблица 4.4

X_1X_2 S_i	00	01	11	10
s_0	$s_1 / 010$	---	$s_1 / 010$	---
s_1	$s_1 / 010$	$s_4 / 001$	$s_0 / 001$	$s_3 / 011$
s_3	$s_0 / 100$	$s_0 / 100$	---	---
s_4	---	$s_0 / 100$	---	---

Таблица 4.5

X_1X_2 S_i	00	01	11	10
s_0	$s_1 / 010$	---	$s_1 / 010$	---
s_1	$s_1 / 010$	$s_2 / 001$	$s_0 / 001$	$s_2 / 011$
s_2	$s_0 / 100$	$s_0 / 100$	---	---

Из рассмотренного примера следует, что абстрактный синтез микропрограммного автомата состоит из выполнения следующих этапов:

- 1) построение графа автомата по микропрограмме, заданной графической схемой алгоритма;
- 2) минимизация числа состояний полностью определенного автомата;
- 3) получение не полностью определенного автомата;
- 4) минимизация числа состояний не полностью определенного автомата.

Структурный этап синтеза микропрограммного автомата не отличается от структурного этапа канонического метода синтеза.

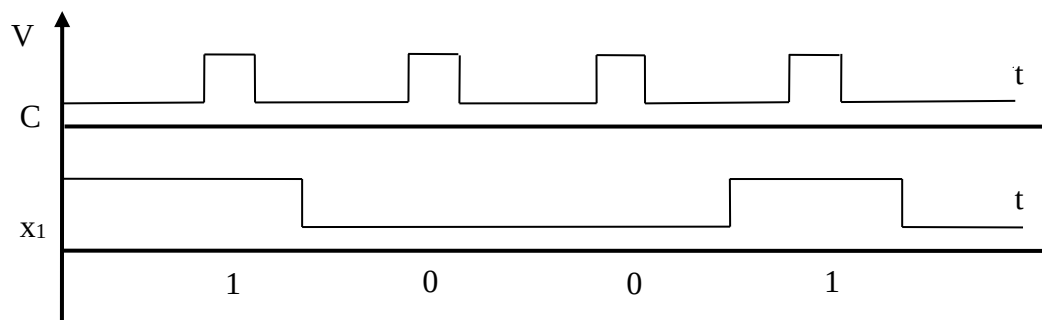
5. АСИНХРОННЫЙ АВТОМАТ

Асинхронным автоматом называется автомат, изменение состояния которого может происходить в произвольный момент времени, и этот момент времени определяется только изменением входного сигнала.

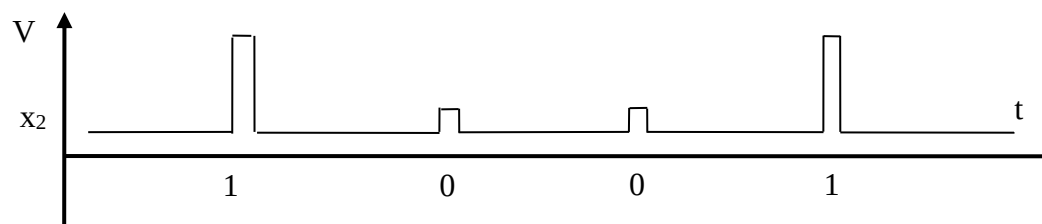
Выбор способа реализации автомата (синхронный или асинхронный) определяется способом представления входного сигнала.

Примеры основных способов представления входных сигналов:

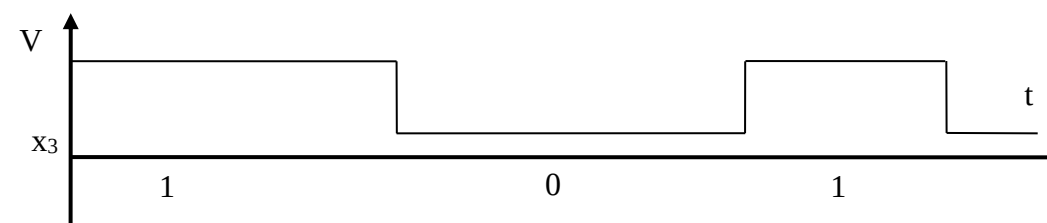
1) потенциальный с синхронизацией;



2) импульсный сигнал



3) потенциальный сигнал



При потенциальном сигнале высокий уровень потенциала соответствует логической единице, низкий — логическому нулю, длительности 1 и 0 могут быть разными.

Первые два способа представления сигнала определяют синхронную реализацию автомата, а третий — асинхронную.

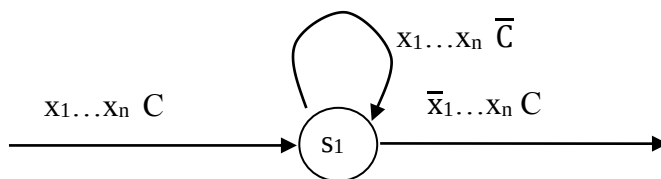


Рис. 5.1

В синхронном автомате сигнал синхронизации C обеспечивает устойчивую работу автомата и учитывается неявно. На рис. 5.1 приведен пример фрагмента графа с явным учетом сигнала синхронизации, на нем при $C = 0$ не происходит изменения текущего состояния автомата.

При асинхронной реализации автомата сигнал синхронизации отсутствует, поэтому для обеспечения устойчивой работы все его состояния должны быть устойчивыми уже на абстрактном этапе синтеза.

Состояние s_i называется устойчивым, если при переходе в это состояние под воздействием входного символа p_k автомат остается в этом состоянии до тех пор, пока на его вход не поступит символ, отличный от p_k (рис. 5.1).

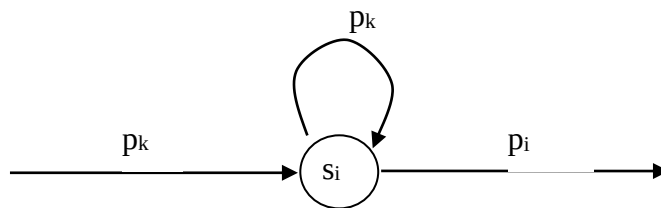


Рис. 5.2

Асинхронным автоматом называется автомат, если уже на абстрактном этапе все его состояния устойчивы.

Пример асинхронного автомата (рис. 5.3):

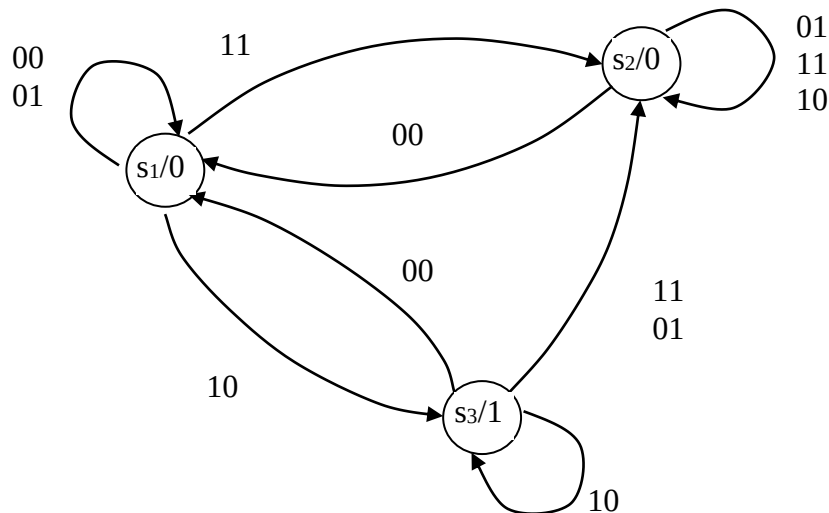


Рис. 5.3

Для представления асинхронного автомата используется модель автомата Мура. Графу автомата Мура (рис. 5.3) соответствует следующая таблица переходов и выходов автомата Мура (табл.5.1). В клетках этой таблицы в скобках заключены устойчивые состояния и для асинхронного автомата важно, в каком из устойчивых состояний находится автомат.

Пусть автомат находится в устойчивом состоянии s_1 воздействием входного сигнала 00 и если входной сигнал изменился на 10, то для определения состояния, в которое переходит автомат необходимо:

- двигаться по строке соответствующей текущему состоянию до столбца, отмеченного новым входным сигналом, в результате определяется состояние, в которое должен перейти автомат;
- двигаться по этому столбцу до устойчивого нового состояния.

Таблица 5.1

w_i	x_1x_2 s_i	00	01	11	10
0	s_1	(s_1)	(s_1)	s_2	s_3
0	s_2	s_1	(s_2)	(s_2)	(s_2)
1	s_3	s_1	s_2	s_2	(s_3)

Двигаясь по строке отмеченной символом текущего состояния s_1 от столбца, отмеченного текущим сигналом 00, до столбца, отмеченного новым входным сигналом 10, определяем новое состояние s_3 . Далее двигаться по столбцу, отмеченного новым входным сигналом 10, до устойчивого нового состояния (s_3).

5.1. СИНТЕЗ АСИНХРОННОГО АВТОМАТА

Синтез асинхронного автомата отличается от синтеза синхронного автомата.

На абстрактном этапе синтеза асинхронного автомата отличие состоит в выполнении следующих дополнительных действий:

- перекодированием входных и выходных символов для обеспечения распознавания соседних одинаковых символов;
- при переходе от алфавитного оператора к графу автомата необходимо сделать все состояния автомата устойчивыми.

Пример необходимости перекодирования входных и выходных символов абстрактного автомата А.

Пусть $P = W = \{a, b, c\}$. Входное слово $a a b b$ автомат перерабатывает в выходное слово $b c c a$. При переходе к структурному автомату символы кодируются: $a = 00$, $b = 11$, $c = 10$.

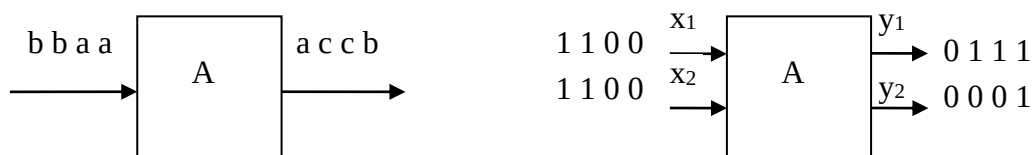


Рис. 5.4

На рис. 5.4 видно, что на уровне сигналов структурного автомата два одинаковых, подряд следующих символа, неразличимы.

Необходимо расширить алфавиты: $P' = W' = \{a, a', b, b', c, c'\}$. Теперь входное слово $a' b' b' a'$ автомат перерабатывает в выходное слово $b' c' c' a'$ (рис. 5.5). При переходе к структурному автомату символы кодируются: $a = 000, a' = 001, b = 110, b' = 111, c = 100, c' = 101$.

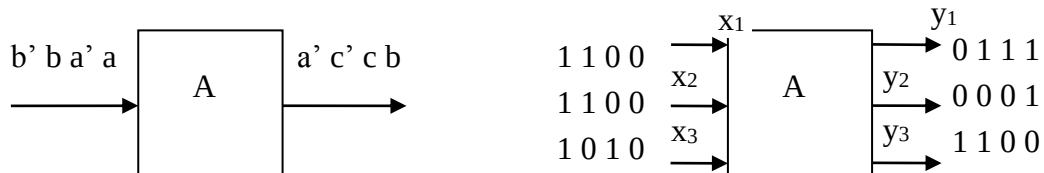


Рис. 5.5

Расширение алфавита требует введение третьего разряда в код символа, но позволяет различать два одинаковых, подряд следующих символа.

На структурном этапе синтеза асинхронного автомата отличие состоит в следующем:

- другой целью кодирования состояний автомата;
- другой целью синтеза комбинационной схемы;
- элементом памяти, используемым в автомате.

Задача кодирования состояний асинхронного автомата состоит в обеспечении устойчивой работы автомата за счет устранения, так называемого явления состязания элементов памяти.

5.2. СОСТЯЗАНИЕ ЭЛЕМЕНТОВ ПАМЯТИ

Реальные элементы памяти обладают определенной задержкой во времени срабатывания, причем величина задержки заранее неизвестна и зависит от технологии изготовления и от внешних факторов в процессе эксплуатации, поэтому определить соотношение задержек различных элементов памяти заранее невозможно.

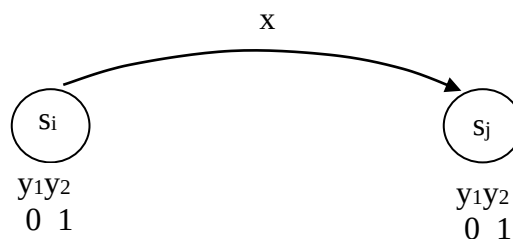


Рис. 5.6

Пусть во фрагменте графа (рис. 5.6) при переходе из s_i в s_j элементы памяти y_1 и y_2 имеют задержки Δ_1 и Δ_2 , между ними возможны следующие соотношения, т.е. если:

- $\Delta_1 = \Delta_2$, то $01 \rightarrow 10$;
- $\Delta_1 > \Delta_2$, то $01 \rightarrow 00 \rightarrow 10$;
- $\Delta_1 < \Delta_2$, то $01 \rightarrow 11 \rightarrow 10$.

Таким образом, наличие различных задержек приводит к тому, что при переходе из s_i в s_j автомат может оказаться в некотором промежуточном незапланированном состоянии, что может привести к неправильной работе автомата.

Это явление называется *состязанием элементов памяти*.

В примере (рис. 5.6) могут быть три варианта соотношений. Первый вариант нереален. Во втором и третьем варианте появляются промежуточные состояния 00 и 11, именно переходы через эти состояния называются *состязанием элементов памяти*.

Состязания могут быть критическими и некритическими. Если состязание приводит к переходу в незапланированное устойчивое состояние, то такое состязание называется *критическим*.

Если же состязание приводит к переходу в незапланированное неустойчивое состояние, а затем и к переходу в нужное устойчивое состояние, то такое состязание называется *некритическим*.

Пример возникновения состязаний в автомате, который задан графом (рис.5.3) и таблицей переходов (табл. 5.1).

Пусть состояния закодированы следующим образом (табл. 5.2) и тогда закодированная таблица переходов имеет вид (табл. 5.3):

Таблица 5.2

s_i	y_1	y_2
s_1	0	0
s_2	0	1
s_3	1	1

Таблица 5.3

z	$\begin{matrix} x_1x_2 \\ y_1y_2 \end{matrix}$	00	01	11	10
0	00	(00)	(00)	01	11
0	01	00	(01)	(01)	(01)
1	11	00	01	01	(11)

Пусть в автомате должен произойти переход по входному сигналу 10 из устойчивого состояния с кодом 00 в устойчивое состояние с кодом 11.

Если $\Delta_1 > \Delta_2$, то $00 \rightarrow 01 \rightarrow 11$.

Состязание критическое, так как из-за такого соотношения между задержками автомат оказывается в промежуточном состоянии 01, которое для автомата является устойчивым и перехода в состояние 11 не произойдет и это приводит к неправильной работе автомата.

Если $\Delta_1 < \Delta_2$, то $00 \rightarrow 10 \rightarrow 11$.

Состязание некритическое, так как из-за такого соотношения между задержками код в блоке памяти становится равным 10, а такой код для

кодирования состояний автомата не используется и автомат далее перейдет в устойчивое состояние с кодом 11.

Так как неизвестно соотношение между задержками элементов памяти и невозможно определить тип состязания: критическое или некритическое, то надо вообще исключить состязания элементов памяти.

5.3. КОДИРОВАНИЕ СОСТОЯНИЙ АСИНХРОННОГО АВТОМАТА

Исключение состязаний является основной целью кодирования состояний асинхронного автомата. Для этого необходимо состояния, между которыми происходит переход, закодировать так, чтобы при этом переходе менялся только один элемент памяти.

Необходимое число разрядов кода q определяется также как при кодировании синхронного автомата, т.е. $q_{\min} = \lceil \log_2 k \rceil$, $q_{\max} = k$, где k – число состояний автомата.

Способ кодирования соседними кодами

Этот способ предполагает использование минимального числа элементов памяти и состоит в том, что состояния, между которыми существует переход, кодируются соседними кодами.

Однако, даже используя код любой длины, не всегда удастся на всех переходах закодировать состояния соседними кодами (рис. 5.7).

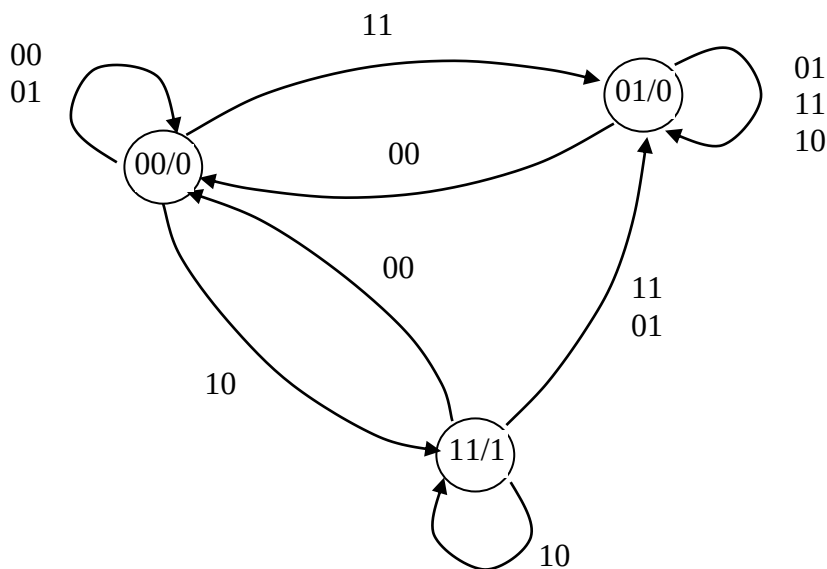


Рис. 5.7

В этом случае необходимо вводить дополнительные промежуточные состояния, которые являются неустойчивым и называются транзитными. Число транзитных состояний на переходе может быть любым. Они служат для исключения состязаний и задания очередности изменения элементов

памяти. На рис. 5.8 между состоянием с кодом 00 и состоянием с кодом 11 введено транзитное состояние с кодом 10.

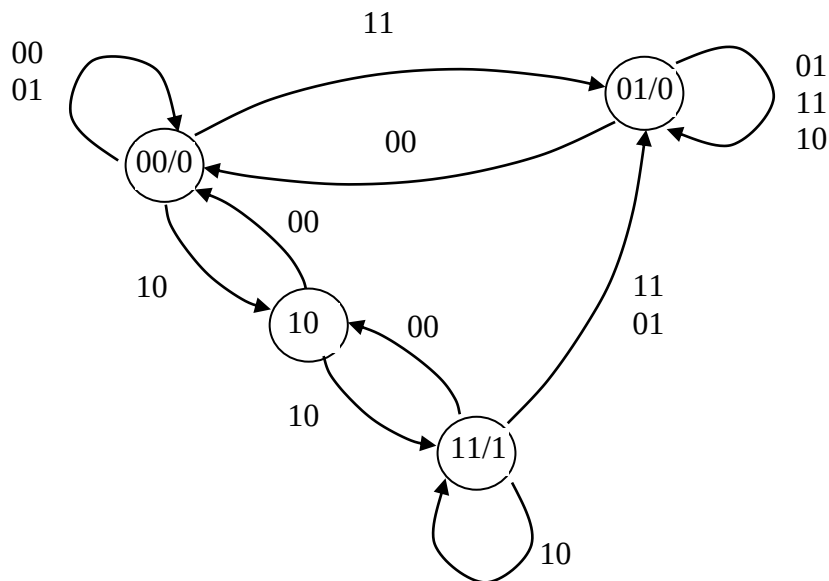


Рис. 5.8

Если число переходов велико и при минимальном числе элементов памяти не удастся соседними кодами закодировать состояния, то необходимо увеличить на единицу число разрядов кода и повторить процедуру кодирования, при этом надо стремиться при минимально возможной длине кода ввести минимальное число транзитных состояний.

Универсальный способ кодирования

Это кодирование состояний унитарным кодом.

В данном способе кодирования состояние s_i кодируется кодом, содержащим одну единицу в i – том разряде.

Для исключения состязаний элементов памяти при переходе из s_i в s_j на этом переходе вводится транзитное состояние, которое кодируется кодом, содержащим две единицы в i и j разрядах (рис. 5.9).

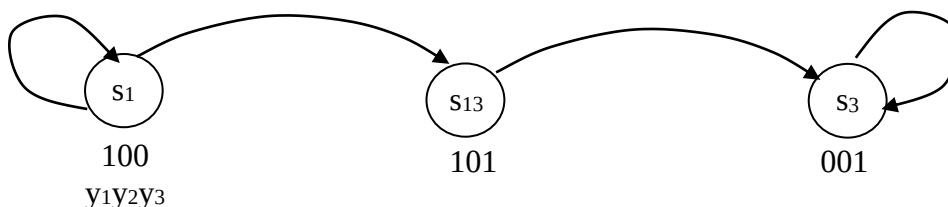


Рис. 5.9

Универсальный способ кодирования позволяет получить функции возбуждения и выхода прямо по графу автомата, которые не требуется минимизировать, но иногда можно упростить.

Пример универсального способа кодирования.

После кодирования унитарным кодом состояний автомата, заданного графом на рис. 5.3: $s_1 = 100$, $s_2 = 010$, $s_3 = 001$ и введения транзитных состояний, получается граф на рис. 5.10.

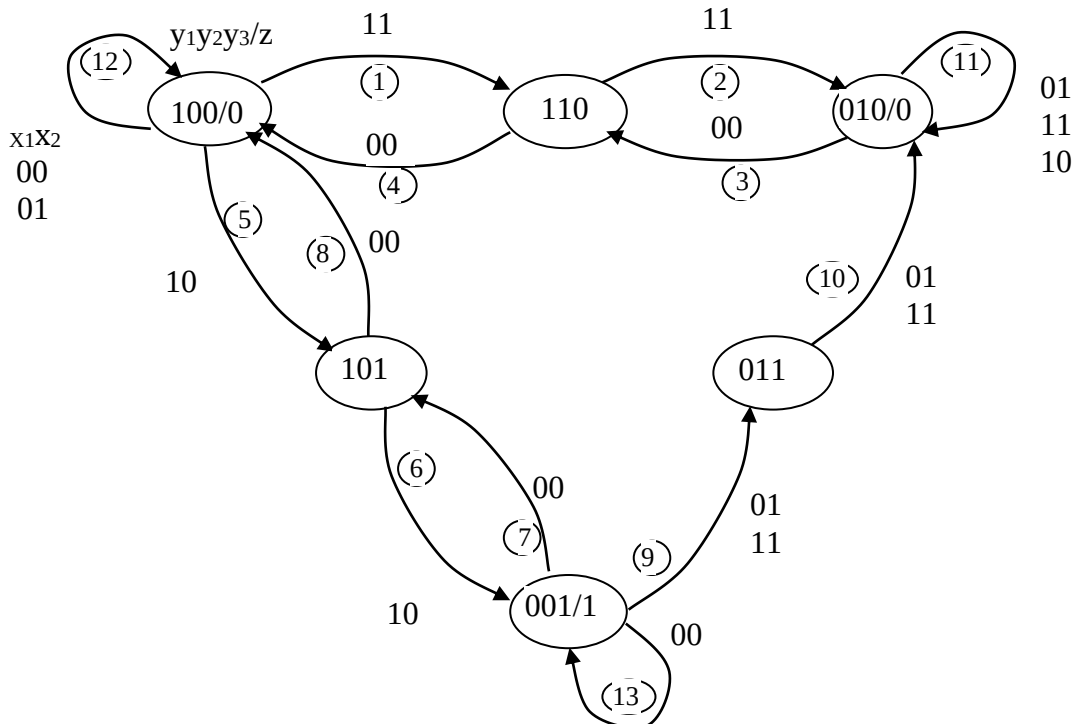


Рис. 5.10

Если в качестве элемента памяти задан D-триггер, то при построении его функций возбуждения необходимо обеспечить его переходы: $0 \rightarrow 1$ и $1 \rightarrow 1$. Например, для реализации двенадцатого перехода (номер перехода на рис. 5.10 обведен) надо в функцию возбуждения первого триггера включить конъюнкцию $y_1 \bar{y}_2 \bar{y}_3$, а для реализации первого перехода из состояния $s_1 = 100$ в транзитное состояние с кодом 110, необходимо второй триггер установить в единицу, для этого в функцию возбуждения второго триггера надо включить конъюнкцию $x_1 x_2 y_1$, как только $y_2 = 1$, конъюнкция $y_1 \bar{y}_2 \bar{y}_3$ примет значение ноль и этим сбросит первый триггер в ноль – так будет реализован второй переход, т.е. из транзитного с кодом 110 автомат перейдет в состояние $s_2 = 010$. Последовательно просматривая все переходы в графе (рис. 5.10), формируются конъюнкции, которые включаются в соответствующие функции возбуждения. Под каждой конъюнкцией указан номер перехода, который она обеспечивает:

$$y'_{D1} = \bar{x}_1 \bar{x}_2 y_2 \underset{(3)}{V} \bar{x}_1 \bar{x}_2 y_3 \underset{(7)}{V} y_1 \bar{y}_2 \bar{y}_3 \underset{(12)}{V};$$

$$y'_{D2} = x_1 x_2 y_1 \underset{(1)}{V} \bar{x}_1 x_2 y_3 \underset{(9)}{V} x_1 x_2 y_3 \underset{(9)}{V} \bar{y}_1 y_2 \bar{y}_3 \underset{(11)}{V};$$

$$z = y_3.$$

Пусть в момент времени t_0 автомат находится в состоянии s_1 , $y_1y_2y_3 = 100$, а входные сигналы $x_1x_2 = 01$. Функции возбуждения триггеров имеют следующие значения $y'_{D1} = 1$, $y'_{D2} = 0$, $y'_{D3} = 0$ и это подтверждает состояние 100, до момента t_1 .

78

При переходе из s_1 в s_2 последовательно меняется вначале y_2 и затем y_1 . Это обеспечивает отсутствие состязаний между ними.

Если в качестве элемента памяти задан RS-триггер, то при построении его функций возбуждения необходимо обеспечить его переходы: $0 \rightarrow 1$ и $1 \rightarrow 0$. Например, для реализации первого перехода (рис. 5.10) из состояния: $s_1 = 100$ в транзитное состояние 110, необходимо второй триггер установить в единицу, для этого в функцию возбуждения входа S второго триггера надо включить конъюнкцию $x_1 x_2 y_1$. Далее для реализации второго перехода из транзитного 110 в состояние $s_2 = 010$ необходимо первый триггер сбросить в ноль, для этого в функцию возбуждения входа R первого триггера надо включить конъюнкцию $x_1 x_2 y_2$. Последовательно просматривая все переходы в графе (рис. 5.10), формируются конъюнкции, которые включаются в соответствующие функции возбуждения:

$$\begin{aligned} y'_{s1} &= \bar{x}_1 \bar{x}_2 y_2 \vee \bar{x}_1 \bar{x}_2 y_3; \\ &\quad \quad \quad (3) \quad \quad \quad (7) \\ y'_{s2} &= x_1 x_2 y_1 \vee \bar{x}_1 x_2 y_3 \vee x_1 x_2 y_3; \\ &\quad \quad \quad (1) \quad \quad \quad (9) \quad \quad \quad (9) \\ y'_{s3} &= x_1 \bar{x}_2 y_1; \\ &\quad \quad \quad (5) \\ y'_{R1} &= x_1 x_2 y_2 \vee x_1 \bar{x}_2 y_3; \\ &\quad \quad \quad (2) \quad \quad \quad (6) \\ y'_{R2} &= \bar{x}_1 \bar{x}_2 y_2; \\ &\quad \quad \quad (4) \\ y'_{R3} &= \bar{x}_1 \bar{x}_2 y_3 \vee \bar{x}_1 x_2 y_3 \vee x_1 x_2 y_3; \\ &\quad \quad \quad (8) \quad \quad \quad (10) \quad \quad \quad (10) \\ z &= y_3. \end{aligned}$$

На рис. 5.12 представлена временная диаграмма первого и второго перехода:

Пусть в момент времени t_0 автомат находится в устойчивом состоянии s_1 , $y_1 y_2 y_3 = 100$, а входные сигналы $x_1 x_2 = 01$. Функции возбуждения триггеров имеют следующие значения: $y'_{R1} = 0$, $y'_{s1} = 0$, $y'_{R2} = 0$, $y'_{s2} = 0$ и это подтверждает состояние 100, до момента t_1 .

В момент t_1 меняется x_1 : $0 \rightarrow 1$ и к моменту t_2 изменяет $y'_{D2} = 1$, а это к моменту времени t_3 устанавливает триггер $y_2 = 1$; в интервале t_3 и t_4 получается транзитное состояние 110. Далее изменение y_2 меняет значение y'_{D1} : $1 \rightarrow 0$, которое в свою очередь устанавливает в момент t_5 триггер $y_1 = 0$, после t_5 автомат находится в состоянии 010, т.е. в состоянии s_2 .

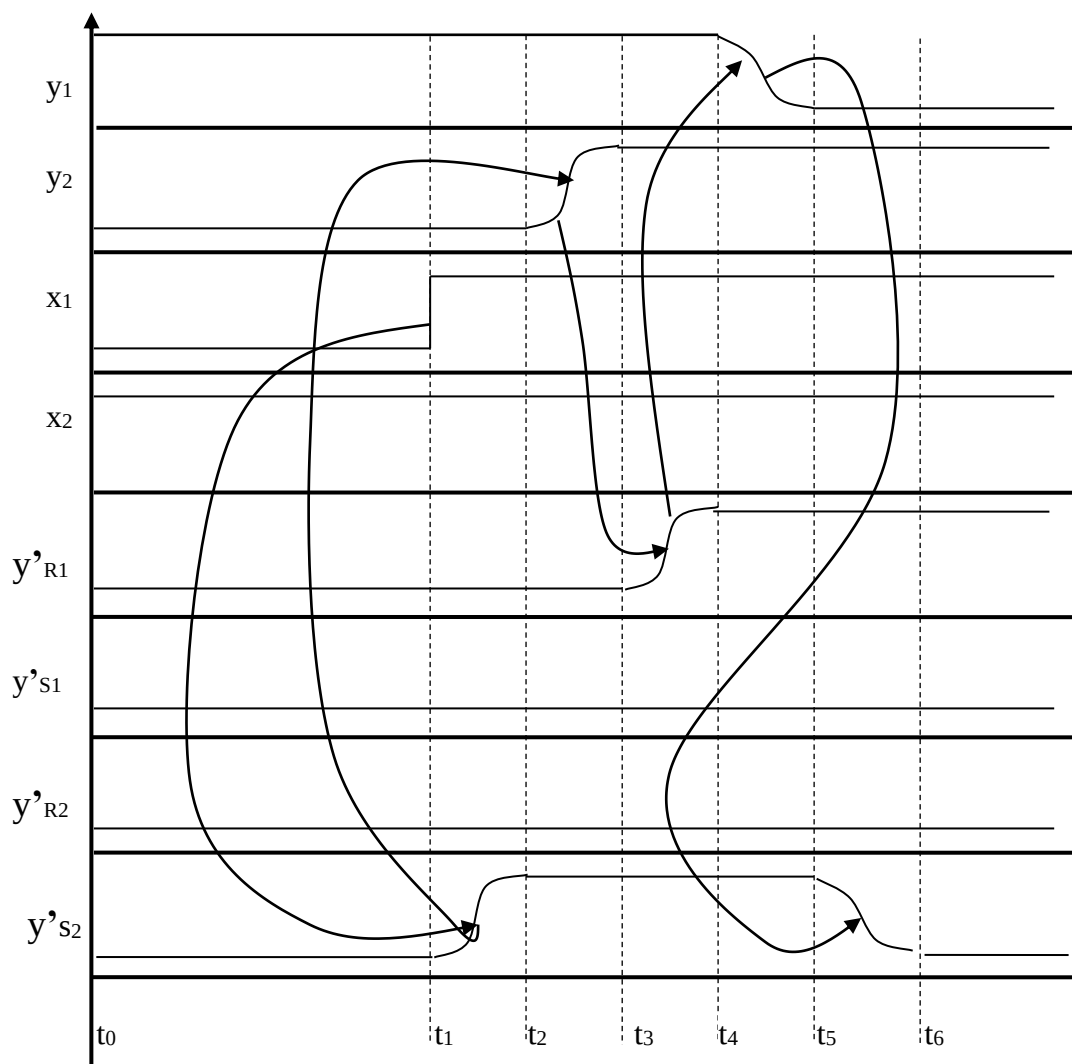


Рис. 5.12

При переходе из s_1 в s_2 последовательно меняется вначале y_2 и затем y_1 . Это обеспечивает отсутствие состязаний между ними.

5.4. СИНТЕЗ СХЕМ ЭЛЕМЕНТОВ ПАМЯТИ

Методы теории автоматов позволяют описывать функционирование и синтезировать логические схемы элементов памяти.

По логическому функционированию элементы памяти (триггеры) делятся на следующие типы: RS, JK, D и T.

По принципу записи и считыванию информации схемы триггеров делятся:

- в зависимости от наличия или отсутствия сигнала синхронизации:
 - а) асинхронные;
 - б) синхронные;
- в зависимости от времени реакции триггера на изменение входного сигнала:
 - а) триггеры без логической задержки;
 - б) триггеры с логической задержкой.

Триггером с логической задержкой называется триггер, который меняет своё состояние по окончании сигнала, вызвавшего это изменение.

На рис. 5.13 -5.16 приведены временные диаграммы работы всех четырех вариантов логических схем RS-триггера.

В триггерах без логической задержки изменение состояния происходит по фронту информационного сигнала (рис. 5.13) или по фронту синхросигнала (рис. 5.14), а в триггерах с логической задержкой по “спаду” информационного сигнала (рис. 5.15) или “спаду” синхросигнала (рис. 5.16).

*Асинхронный RS-триггер
без логической задержки*

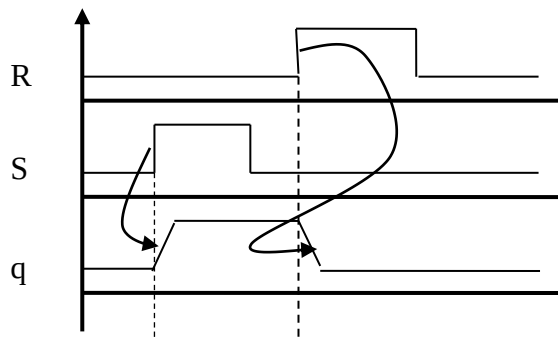


Рис. 5.13

*Синхронный RS-триггер
без логической задержки*

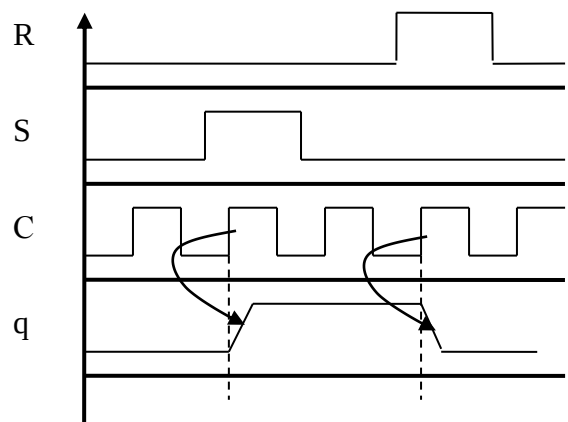


Рис. 5.14

*Асинхронный RS-триггер
с логической задержкой*

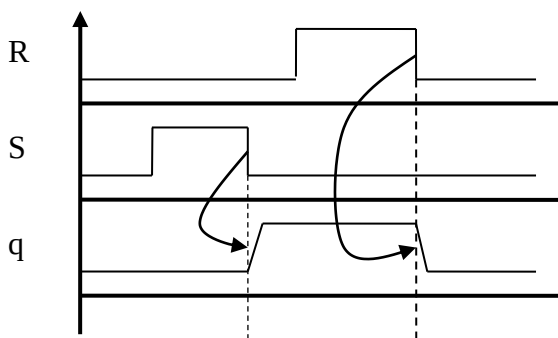


Рис. 5.15

*Синхронный RS-триггер
с логической задержкой*

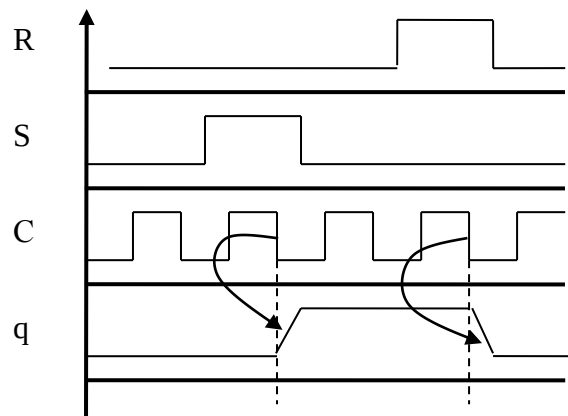


Рис. 5.16

Построение асинхронного RS-триггера без логической задержки из логических элементов потенциального типа.

RS-триггер из логических элементов ИЛИ-НЕ.

На рис. 5.17 приведена схема так называемой “бистабильной ячейки” из элементов ИЛИ-НЕ, которая может находиться в одном из двух состояний. Пусть на одном из выходов схемы, который обозначим – q , значение логический ноль, а на другом, который обозначим – $\bar{q}$, значение логической единицы. После подачи на входы схемы тестовых последовательностей и построения временной диаграммы (рис. 5.18) видно, что это схема является RS-триггером, так как подача 0 0 на входы схемы не меняет значения на её выходах, а подача на входы 1 1 дает на выходах 0 0, т.е. схема перестает быть триггером и следовательно набор 1 1 не допустим. Верхний вход является R-входом, а нижний S-входом. Условное обозначение триггера (рис. 5.19).

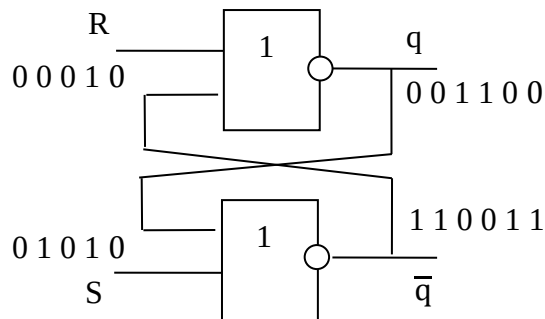


Рис. 5.17

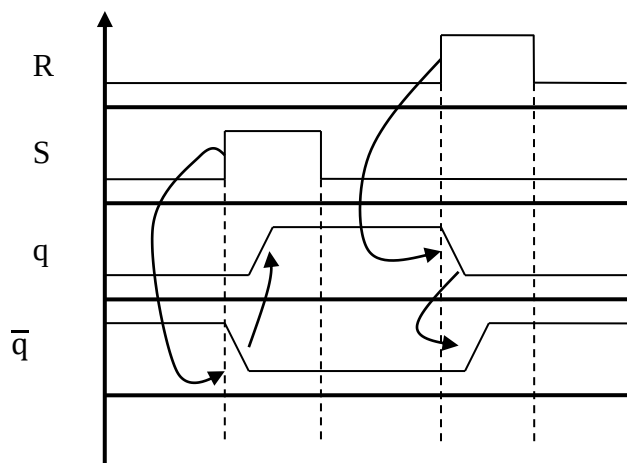


Рис. 5.18

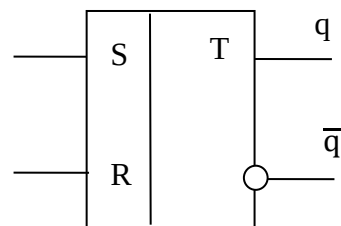


Рис. 5.19

RS-триггер из логических элементов И-НЕ

На рис. 5.20 приведена схема “бистабильной ячейки” из элементов И-НЕ, которая может находиться в одном из двух состояний. Пусть на одном из выходов схемы, который обозначим – q , значение логический ноль, а на другом, который обозначим – $\bar{q}$, значение логической единицы. После подачи на входы схемы тестовых последовательностей и построения временной диаграммы (рис. 5.21) видно, что это схема является RS-триггером с инверсными входами? Так как подача 0 0 на входы схемы дает на выходах 1 1, т.е. схема перестает быть триггером и следовательно набор 0 0 недопустим, а входной набор 1 1 не меняет значения на её выходах. Верхний вход является $\bar{S}$ -входом, а нижний $\bar{R}$ -входом. Условное обозначение триггера (рис. 5.22).

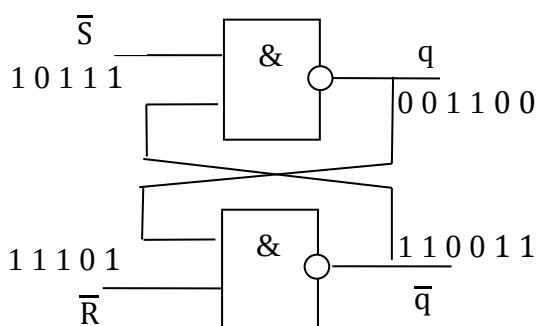


Рис. 5.20

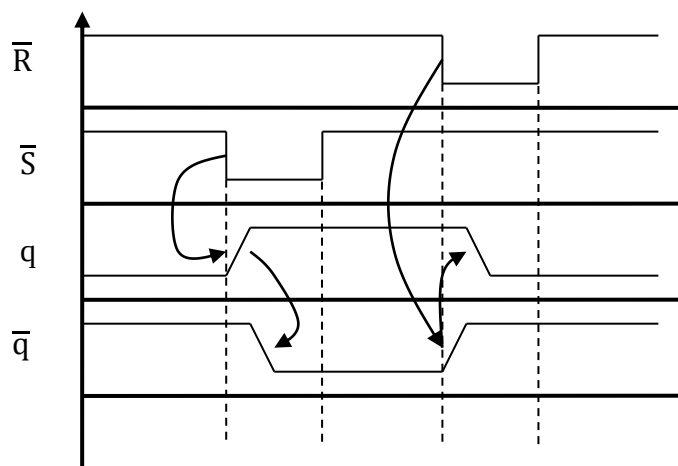


Рис. 5.21

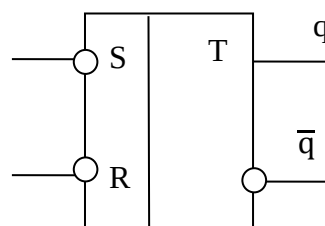


Рис. 5.22

Элементы памяти с установочными входами

Для того чтобы перед началом работы схема триггера находилась в нужном состоянии, в схему вводятся дополнительные так называемые установочные входы R_0 и S_0 для элементов ИЛИ-НЕ (рис. 5.23), $\bar{S}_0$ и $\bar{R}_0$ для элементов И-НЕ (рис. 5.24). Они отличаются от информационных входов только тем, что нужные значения подаются на входы только перед началом работы триггера. После установки триггера в требуемое состояние они меняются на значение, не влияющие на дальнейшую работу триггера. На рис. 5.25 и рис. 5.26 указаны сигналы, устанавливающие триггеры в нулевое состояние, т.е. $q = 0$. На рис. 5.27 и рис. 5.28 даны условные обозначения триггеров с установочными входами.

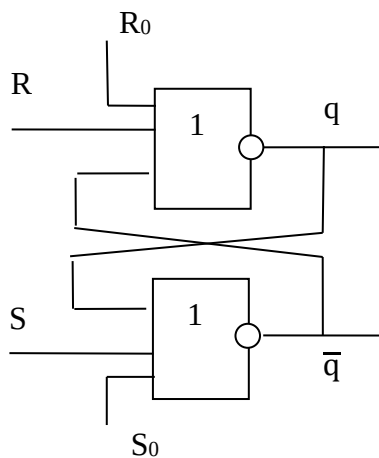


Рис. 5.23

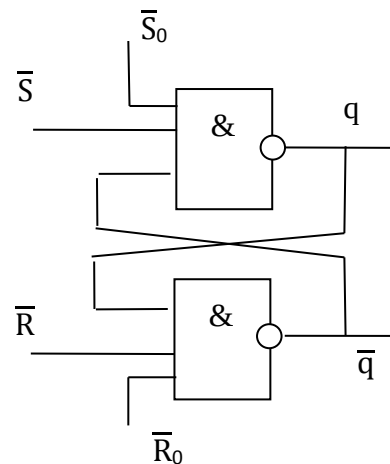


Рис. 5.24

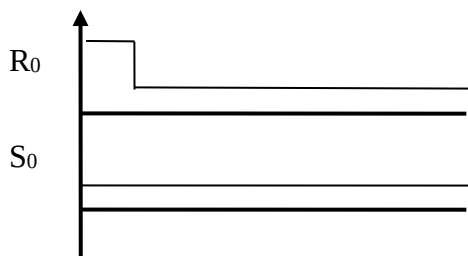


Рис. 5.25

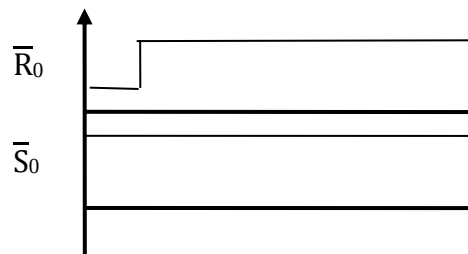


Рис. 5.26

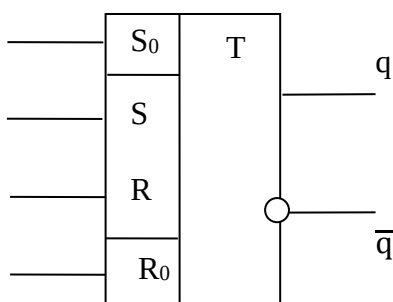


Рис. 5.27

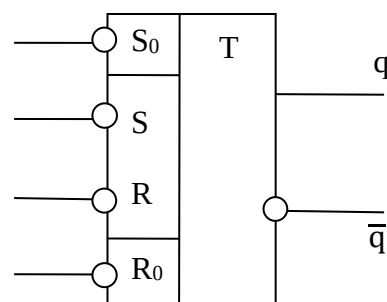


Рис. 5.28

Синхронные элементы памяти.

Синхронизация позволяет:

- исключить влияние временных смещений сигналов триггера на его разных информационных входах;
- позволяет обеспечить одновременное изменение состояний различных элементов памяти.

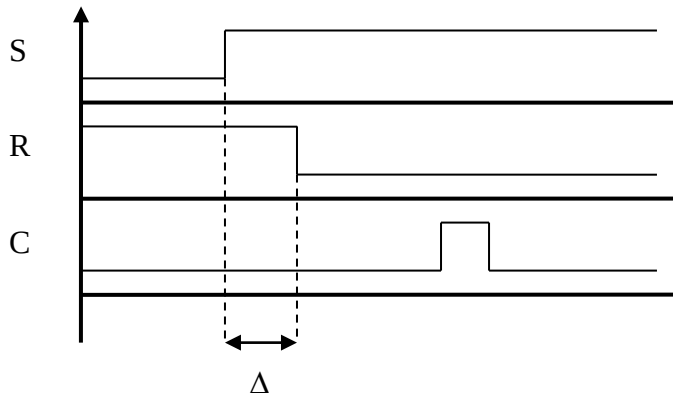


Рис. 5.29

На рис. 5.29 приведен пример исключения временного смещения входных сигналов на величину Δ . Для асинхронного триггера это привело бы к недопустимой ситуации – подачи на входы RS-триггера двух единиц одновременно. Введение сигнала синхронизации позволяет это исключить.

Требования, предъявляемые к сигналу синхронизации:

- сигнал синхронизации по длительности должен быть короче информационного сигнала;
- сигнал синхронизации должен быть во времени сдвинут от изменения информационного сигнала на величину Δ , обеспечивающую устойчивую работу триггера (рис. 5.30).

Информационный сигнал не должен меняться при $C = 1$.

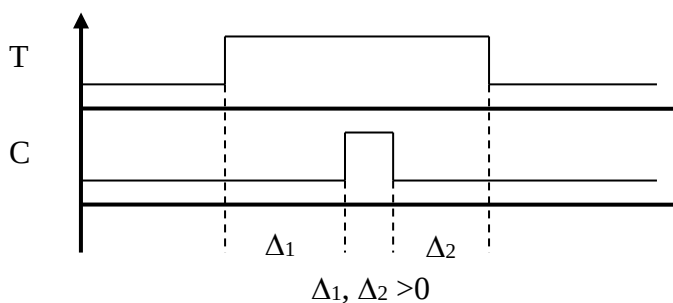


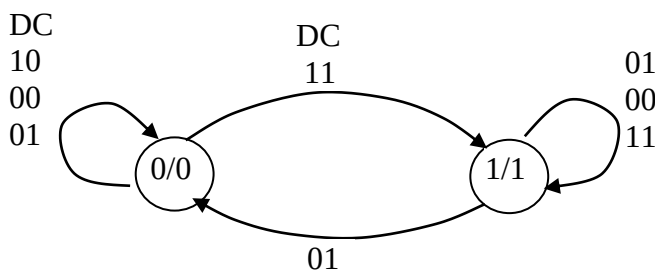
Рис. 5.30

Синтез синхронного D-триггера без логической задержки

При синтезе всех триггерных схем синтезируемое устройство рассматривается как асинхронный автомат Мура, который реализуется из заданных логических элементов с использованием в качестве элемента памяти асинхронный RS-триггер без логической задержки (базовый RS-триггер) из заданных логических элементов.

На рис. 5. 31 граф и таблица переходов автомата Мура (табл. 5.4), задающих синхронный D-триггер

Таблица 5.4



DC q \ DC	00	01	11	10
0	0	0	1	0
1	1	0	1	1

Рис. 5.31

На рис. 5. 32 изображена структурная схема синтезируемого D-триггера. В качестве элемента памяти автомата используется базовый RS-триггер.

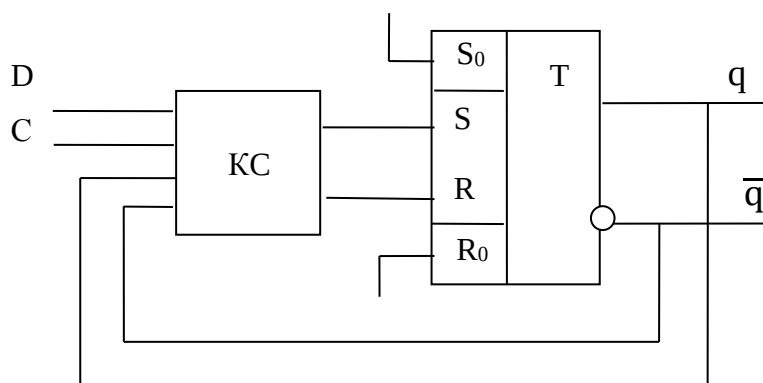


Рис. 5.32

$q(t) \rightarrow q(t+1)$	S	R
0 $\rightarrow$ 0	0	*
0 $\rightarrow$ 1	1	0
1 $\rightarrow$ 0	0	1
1 $\rightarrow$ 1	*	0

Рис. 5.33

Таблица 5.5

DC q \ DC	00	01	11	10
0	0*	0*	10	0*
1	*0	01	*0	*0

По таблице переходов (табл. 5.4), используя матрицу переходов RS-триггера (рис. 5. 33), строится таблица функций возбуждения (табл. 5.5).

Далее по этой таблице строятся карты Карно для функций возбуждения: y'_S (табл. 5.6) и y'_R (табл. 5.7). После минимизации получаются следующие выражения для функций возбуждения RS-триггера в базе ИЛИ-НЕ:

$$y'_S = DC = \overline{\overline{D}\overline{C}} = \overline{D} \vee \overline{C}; \quad y'_R = \overline{D}C = \overline{\overline{\overline{D}\overline{C}}} = \overline{D} \vee \overline{C}.$$

Таблица 5.6

y'_s					D
					C
	0	0	1	0	
q	*	0	*	*	

Таблица 5.7

y'_R					D
					C
	*	*	0	*	
q	0	1	0	0	

На рис. 5.34 приведена схема синтезированного синхронного D-триггера из элементов ИЛИ-НЕ, а на рис. 5.35 его условное обозначение:

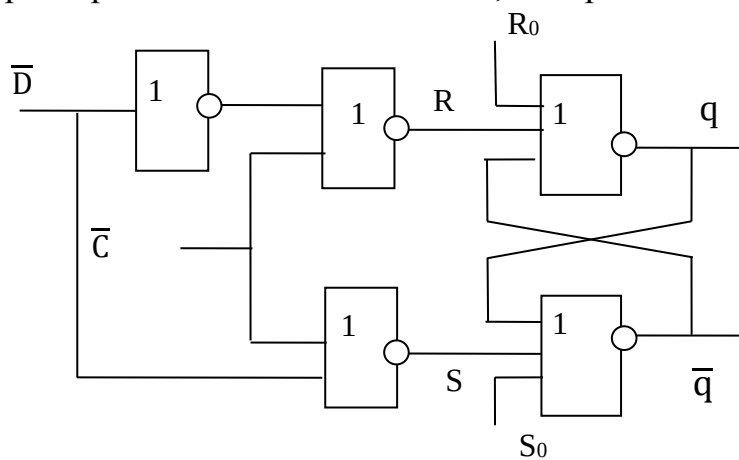


Рис. 5.34

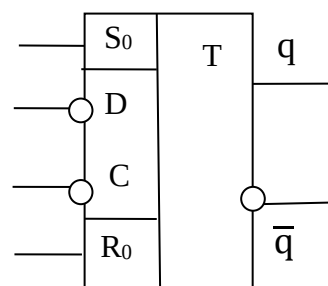


Рис. 5.35

На рис. 5.36 приведена схема синтезированного синхронного D-триггера из элементов И-НЕ, а на рис. 5.37 его условное обозначение:

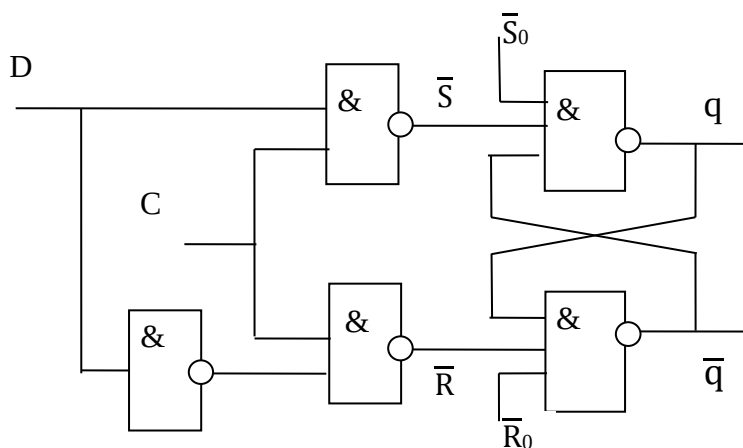


Рис. 5.36

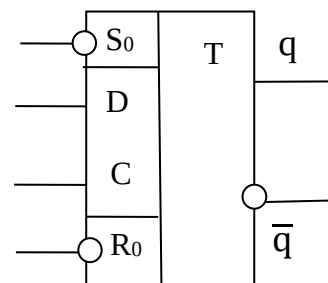


Рис. 5.37

На рис. 5.38 приведена временная диаграмма работы синхронного D-триггера из элементов ИЛИ-НЕ (рис. 5.34).

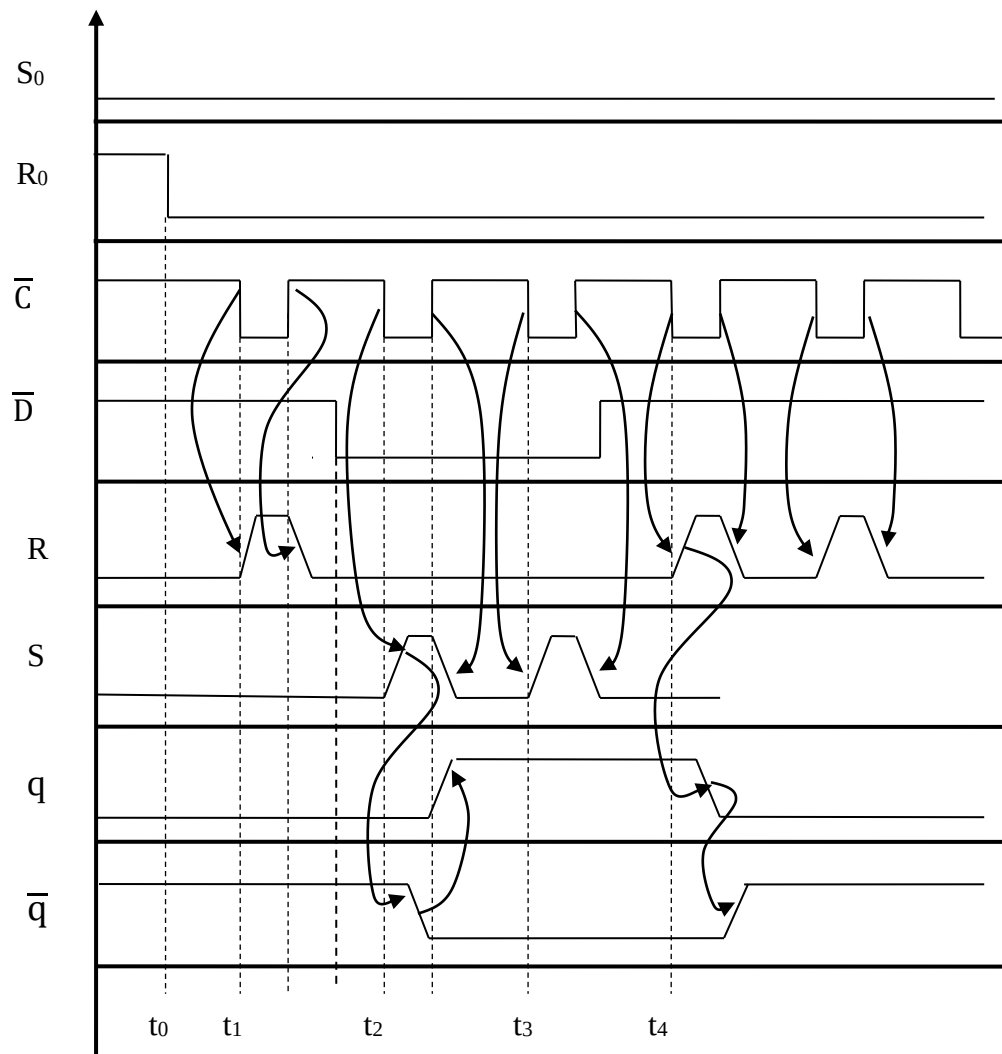


Рис. 5.38

В момент времени t_0 сигнал R_0 , ранее установивший в RS-триггер в 0, становится $R_0 = 0$ и RS-триггер начинает реагировать на сигналы, подаваемые на информационные входы.

В момент t_1 появляется сигнал $R = 1$, однако триггер в состоянии 0 и следовательно смены состояния не происходит.

В момент t_2 появляется сигнал $S = 1$, который устанавливает $\bar{q} = 0$, а затем $q = 1$. Повторный сигнал $S = 1$ в момент t_3 лишь подтверждает состояние $q = 1$.

В момент времени t_4 появившийся сигнал $R = 1$ устанавливает $q = 0$, а затем $\bar{q} = 1$, т.е. сбрасывает триггер в 0.

Синтез асинхронного T- триггера с задержкой.

Триггером с логической задержкой называется триггер, который меняет своё состояние по окончании сигнала, вызвавшего это изменение, т.е. триггер меняет состояние по спаду информационного сигнала (рис. 5.39).

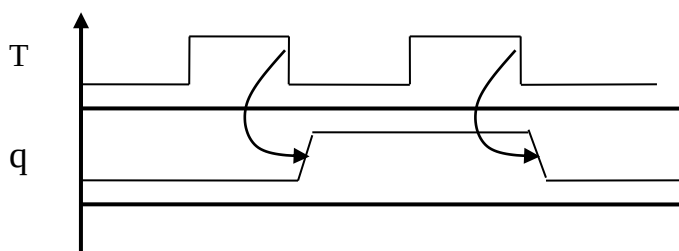


Рис. 5.39

Появление входного информационного сигнала $T = 1$ устанавливает новое состояние триггера $q = 1$, однако на выходе это значение появляется лишь после исчезновения сигнала $T = 1$. Эффект такой логической задержки можно организовать, используя два элемента памяти, т.е. для задания такого триггера используется автомат Мура, который имеет четыре состояния и соответственно граф – четыре вершины (рис. 5.40):

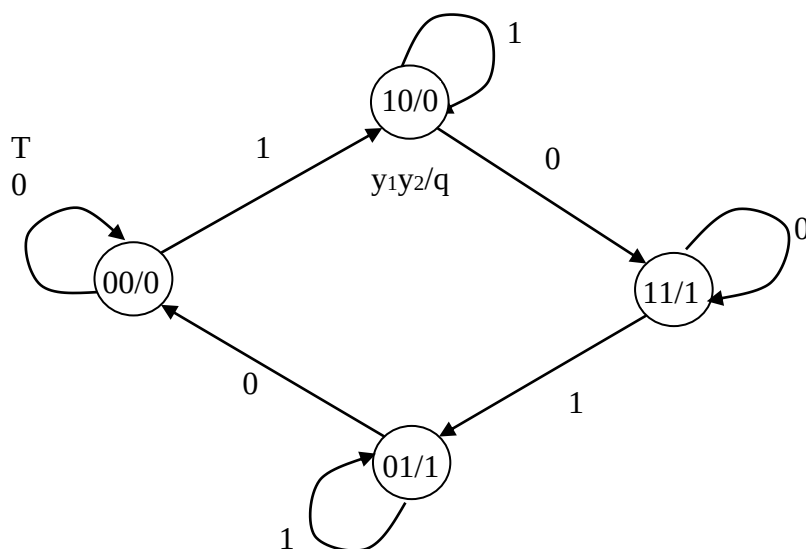


Рис. 5.40

Таблица переходов автомата (табл.5.8) и его структурная схема (рис.5.41):

Таблица 5.8

q	T	0	1
	y ₁ y ₂		
0	00	00	10
0	10	11	10
1	11	11	01
1	01	00	01

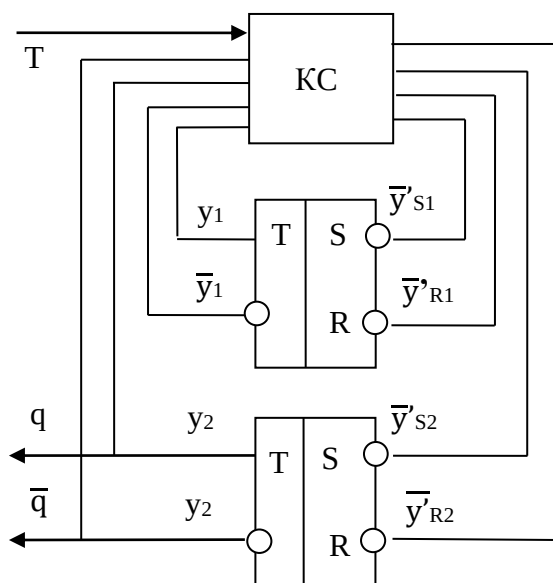


Рис.5.41

По таблице переходов (табл. 5.8), используя матрицу переходов RS-триггера (рис. 5. 42), строится таблица функций возбуждения (табл. 5.9).

Далее по этой таблице строятся карты Карно функций возбуждения y'_{s1} (табл. 5.10), y'_{r1} (табл. 5.11) и y'_{s2} (табл. 5.12), y'_{r2} (табл. 5.13).

Таблица 5.9

$q(t) \rightarrow q(t+1)$	S R				
$0 \rightarrow 0$	0 *	q \backslash T y ₁ y ₂		0	1
$0 \rightarrow 1$	1 0				
$1 \rightarrow 0$	0 1				
$1 \rightarrow 1$	* 0				
		0	00	0*, 0*	0*, 10
		0	01	10, *0	0*, *0
		1	11	*0, *0	*0, 01
		1	10	01, 0*	*0, 0*

Рис. 5. 42

Далее по этой таблице строятся карты Карно функций возбуждения: y'_{s1} (табл. 5.10), y'_{r1} (табл. 5.11) и y'_{s2} (табл. 5.12), y'_{r2} (табл. 5.13).

Таблица 5.10

y'_{s1} — T

0	0
1	0
*	*
0	*

y₁y₂

Таблица 5.11

y'_{r1} — T

*	*
0	*
0	0
1	0

y₁y₂

Таблица 5.12

y'_{s2} — T

0	1
*	*
*	0
0	0

y₁y₂

Таблица 5.13

y'_{r2} — T

*	0
0	0
0	1
*	*

y₁y₂

После минимизации получаются следующие выражения для функций возбуждения RS-триггеров:

$$y'_{s1} = y_2 \bar{T}, \quad y'_{r1} = \bar{y}_2 \bar{T}, \quad y'_{s2} = \bar{y}_1 T, \quad y'_{r2} = y_1 T.$$

Для построения схемы Т-триггера в базисе И-НЕ в качестве элемента памяти используется базовый RS-триггер с инверсными входами. Выражения инверсных функций возбуждения RS-триггеров имеют вид:

$$\overline{y'_{s1}} = \overline{y_2 \bar{T}}, \quad \overline{y'_{r1}} = \overline{\bar{y}_2 \bar{T}}, \quad \overline{y'_{s2}} = \overline{\bar{y}_1 T}, \quad \overline{y'_{r2}} = \overline{y_1 T}.$$

Схема Т-триггера на логических элементах И-НЕ (рис. 5. 43):

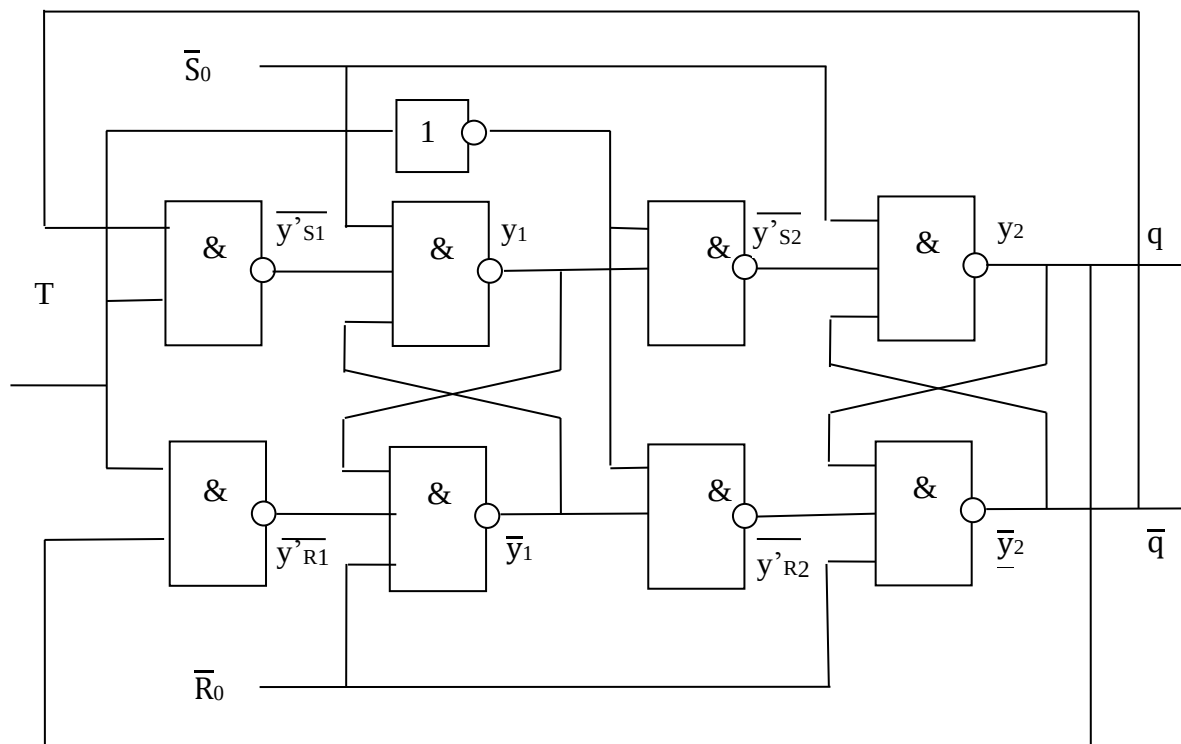


Рис. 5. 43

Такая схема триггера называется двухступенчатой или MS – схемой (Master – Slave). Комбинационная схема, реализующая полученные функции возбуждения, распределена между входной и выходной ступенями триггера, что общепринято при изображении триггерных схем.

Условное обозначение такого Т-триггера (рис. 5. 44). Две буквы Т означают, что схема состоит из двух последовательно соединенных триггеров.

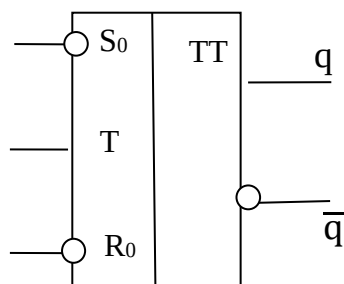


Рис. 5. 44

На рис. 5.45 приведена временная диаграмма работы асинхронного Т-триггера с логической задержкой.

В момент времени t_0 сигнал $\bar{R}_0$, ранее установивший оба RS-триггера в 0, становится $\bar{R}_0 = 1$ и Т-триггер может реагировать на сигнал, подаваемый на информационный вход.

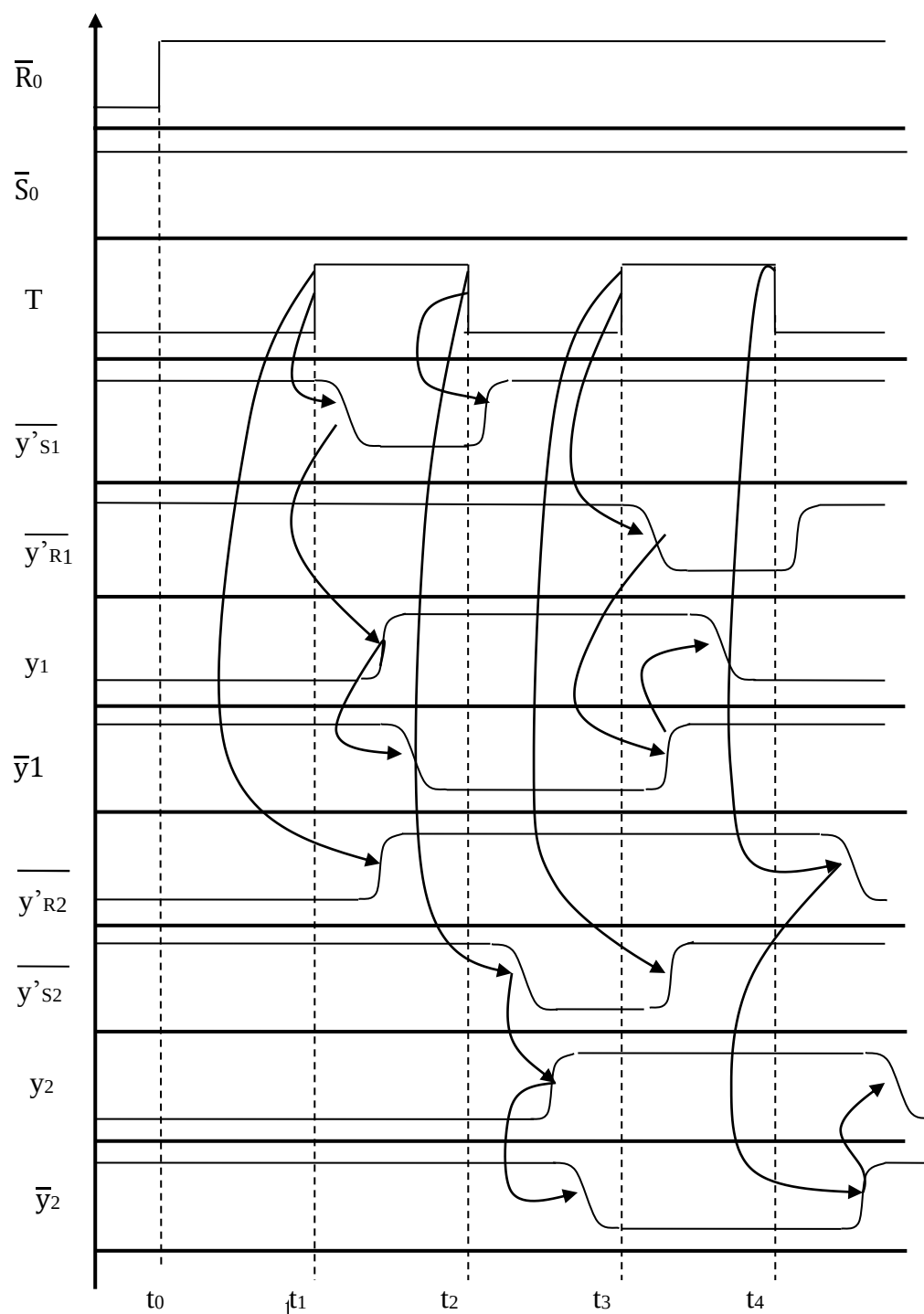


Рис. 5. 45

В момент t_1 появляется сигнал $T = 1$, который меняет значение $\overline{y}'_{s1}$ с 1 на 0 и это устанавливает RS-триггер первой (входной) ступени в единичное состояние, т.е. $y_1 = 1$, а затем $\overline{y}_1 = 0$. RS-триггер второй (выходной) ступени при этом не меняется, так как $\overline{T} = 0$ и $\overline{y}'_{r2} = \overline{y}'_{s2} = 1$.

В момент t_2 появляется входной сигнал $T = 0$ и меняется функция: $\overline{y}'_{s2} = 0$. Это значение устанавливает RS-триггер второй (выходной) ступени в единичное состояние, т.е. $y_2 = 1$, а затем $\overline{y}_2 = 0$. Эти выходы определяют состояние T-триггера, которое изменилось по спаду сигнала T , реализуя тем самым логическую задержку на время сигнала $T = 1$.

Синтез синхронного RS-триггера с логической задержкой.

В синхронном триггере с логической задержкой триггер меняет своё состояние по окончании сигнала, вызвавшего это изменение, т.е. триггер меняет состояние по спаду сигнала синхронизации (рис. 5.46).

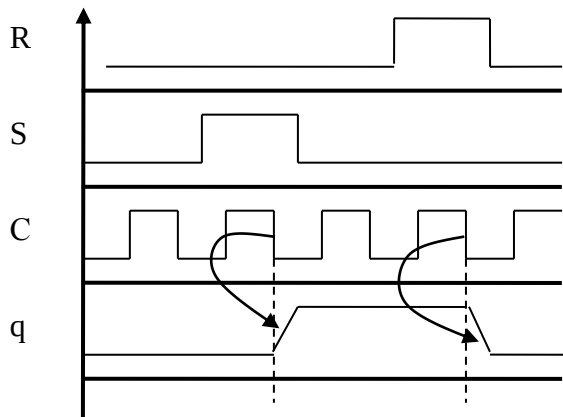


Рис. 5.46

Для реализации логической задержки используется двухступенчатая схема, а для задания используется автомат Мура, который имеет четыре состояния и соответственно граф – четыре вершины (рис. 5.47):

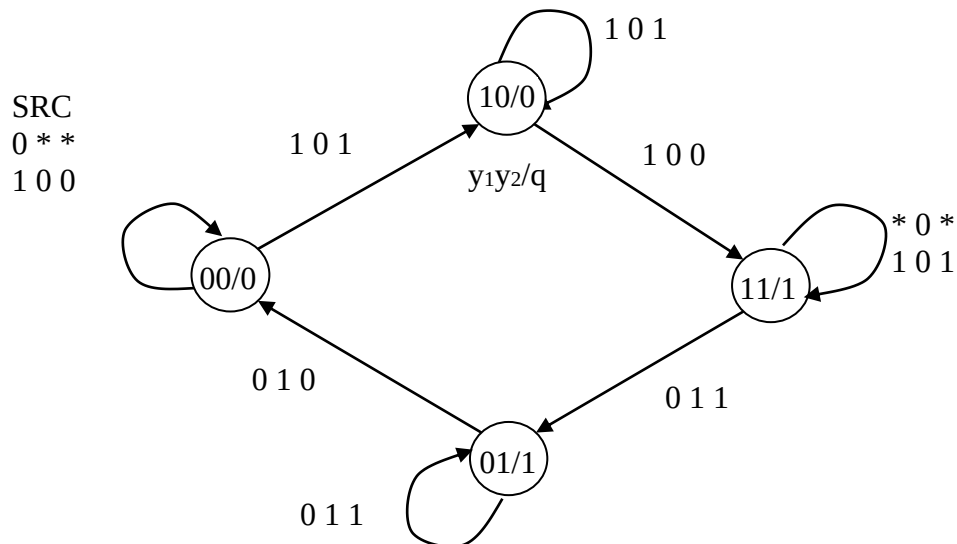


Рис. 5.47

Таблица 5.14

q	SRC y ₁ y ₂	000	001	011	010	110	111	101	100
0	00	00	00	00	00	--	--	10	00
0	10	--	--	--	--	--	--	10	11
1	11	11	11	01	11	--	--	11	11
1	01	--	--	01	00	--	--	--	--

Таблица переходов автомата (табл.5.14) и его структурная схема (рис.5.48):

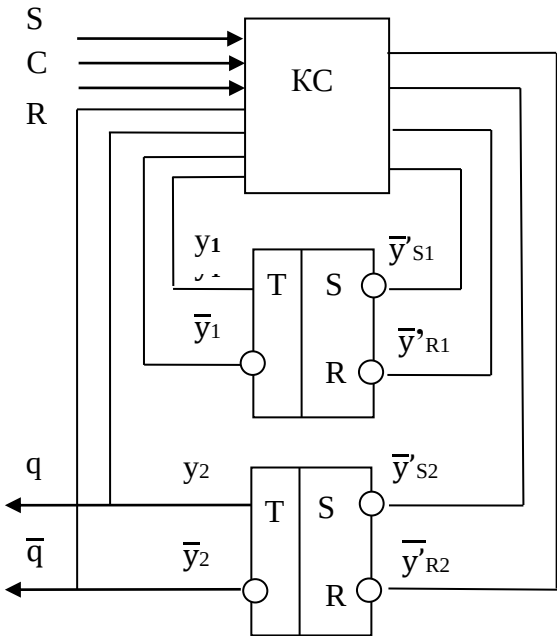


Рис.5.48

$q(t) \rightarrow q(t+1)$	S	R
$0 \rightarrow 0$	0	*
$0 \rightarrow 1$	1	0
$1 \rightarrow 0$	0	1
$1 \rightarrow 1$	*	0

Рис.5.49

По таблице переходов (табл. 5.14), используя матрицу переходов RS-триггера (рис. 5. 49), строятся карты Карно функций возбуждения (без построения таблицы функций возбуждения):
 y'_{s1} (табл. 5.15), y'_{r1} (табл. 5.16) и y'_{s2} (табл. 5.17) , y'_{r2} (табл. 5.18)

Таблица 5.15

y'_{s1}					S		
					R		
					C		
	0	0	0	0	--	1	0
	--	--	--	--	--	*	*
	*	*	0	*	--	*	*
	--	--	0	0	--	--	--
y_1							
y_2							

После минимизации получаются следующие выражения для функций возбуждения RS-триггеров:

$$y'_{S1} = S C, \quad y'_{R1} = R C, \quad y'_{S2} = y_1 \bar{C}, \quad y'_{R2} = \bar{y}_1 \bar{C}.$$

Для построения схемы синхронного RS-триггера с логической задержкой в базисе И-НЕ в качестве элемента памяти используется базовый RS-триггер с инверсными входами из элементов И-НЕ. Выражения инверсных функций возбуждения RS-триггеров имеют вид:

$$\overline{y'_{S1}} = \overline{S C}, \quad \overline{y'_{R1}} = \overline{R C}, \quad \overline{y'_{S2}} = \overline{y_1 \bar{C}}, \quad \overline{y'_{R2}} = \overline{\bar{y}_1 \bar{C}}.$$

Функциональная схема синтезированного двухступенчатого RS-триггера и его условное обозначение приведены на рис.5. 50, 5.51

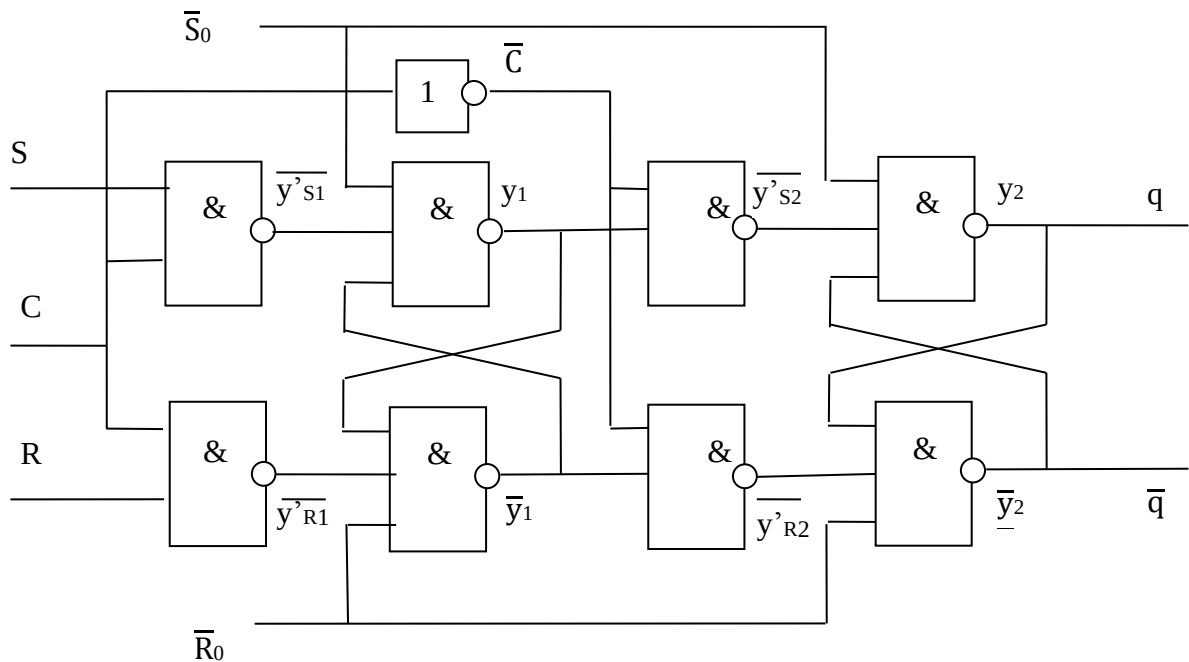


Рис.5. 50

Условное обозначение двухступенчатого RS-триггера с инверсными входами из элементов ИЛИ-НЕ приведено на рис.5. 52.

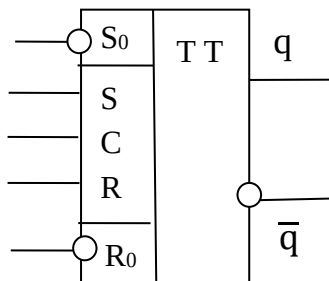


Рис.5. 51

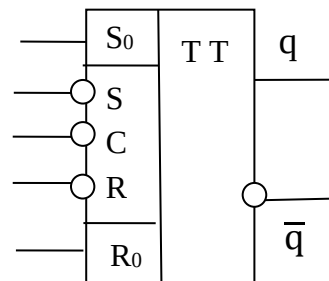


Рис.5. 52

На условном обозначении входы S_0 и R_0 отделены горизонтальной чертой от остальных входов. Это означает, что влияние синхросигнала на них не распространяется.

Временная диаграмма работы двухступенчатого RS-триггера элементов И-НЕ приведена на рис. 5.53.

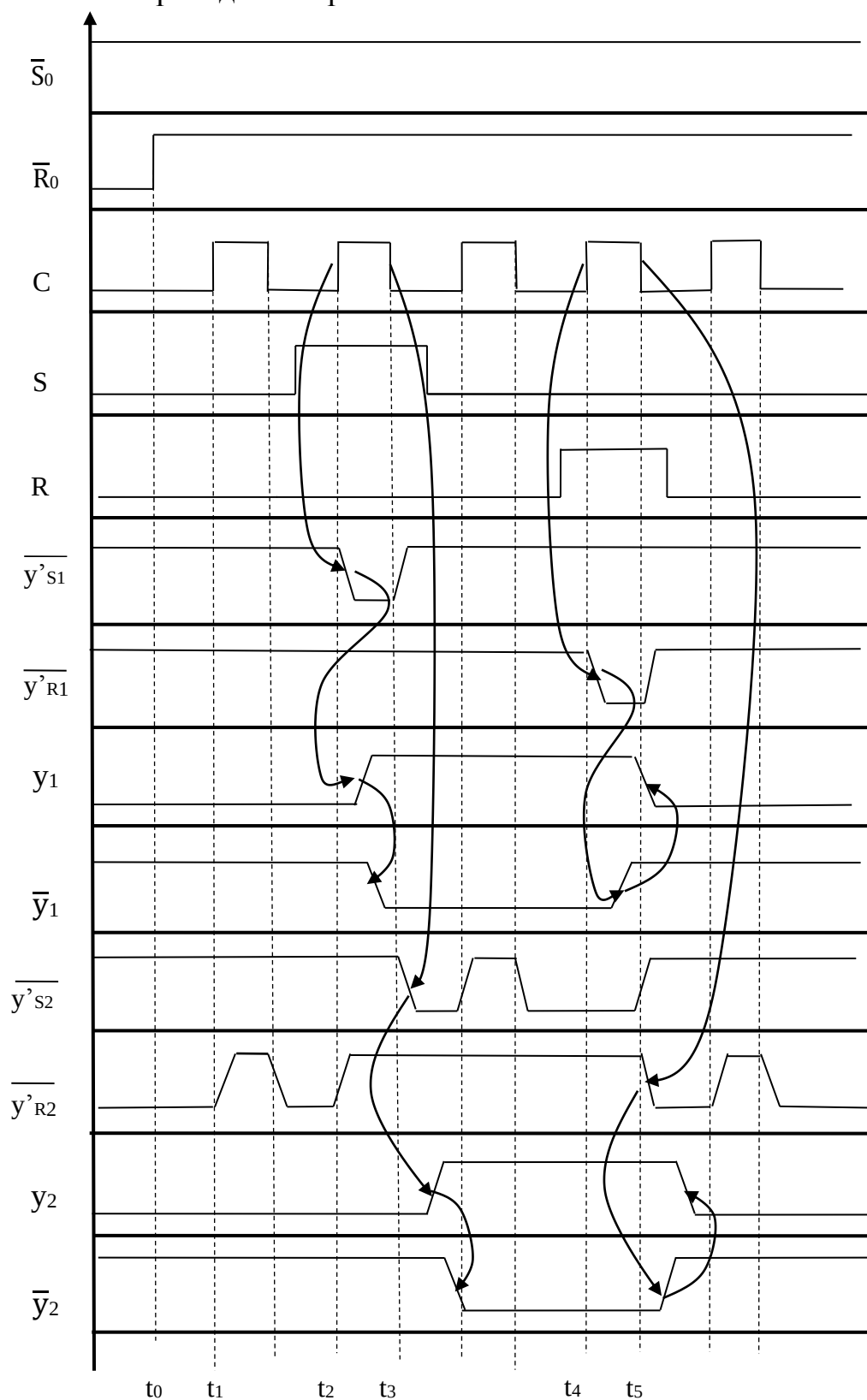


Рис. 5.53

Временная диаграмма строится с учетом временных задержек Δ , вносимых логическими элементами. Стрелками показывается причинно-следственная связь между изменениями состояний логических элементов.

В момент времени t_0 сигнал $\bar{R}_0$, ранее установивший оба RS-триггера в 0, становится $\bar{R}_0 = 1$ и триггер может реагировать на сигналы, подаваемые на его входы. В момент t_1 появляется сигнал $C = 1$, который открывает входы триггера, но так как $S = 0$ и $R = 0$, то нулевые значения функций возбуждения не меняются

В момент t_2 при $S = 1$ по фронту сигнала C меняется значение y'_{s1} с 1 на 0 и это устанавливает RS-триггер первой (входной) ступени в единичное состояние, т.е. $y_1 = 1$, а затем $\bar{y}_1 = 0$. RS-триггер второй (выходной) ступени при этом не меняется, так как $C = 0$, а это определяет значение функций возбуждения $\bar{y}'_{R2} = \bar{y}'_{S2} = 1$.

В момент t_3 при $S = 1$ по спаду сигнала C меняется значение функция: $\bar{y}'_{S2}$ 1 на 0. Это значение устанавливает RS-триггер второй (выходной) ступени в единичное состояние, т.е. $y_2 = 1$, а затем $\bar{y}_2 = 0$. Эти выходы определяют состояние RS –триггера, которое изменилось по спаду сигнала C , реализуя логическую задержку на время сигнала $C = 1$.

Минимально необходимая длительность сигнала синхронизации $C = 1$ должна быть $T_1 = 3 \Delta$, а минимально необходимая длительность сигнала синхронизации $C = 0$ должна быть $T_0 = 4 \Delta$, где Δ - время задержки одного логического элемента.

Синхронные триггеры с логической задержкой используются для реализации “двойной памяти” автомата, обеспечивающей его устойчивую работу.

5.5. ИСКЛЮЧЕНИЕ СОСТЯЗАНИЙ ЭЛЕМЕНТОВ ПАМЯТИ В СИНХРОННОМ АВТОМАТЕ

Явление состязания элементов памяти рассматривалось в работе асинхронного автомата. Устранение данного явления в синхронном автомате можно сделать следующими двумя способами:

- 1) за счет выбора минимально необходимой длительности сигнала синхронизации;
- 2) использованием структуры автомата с двойной памятью.

Первый способ был рассмотрен ранее в примере структурного этапа канонического метода синтеза синхронного автомата, но обычно на практике этот способ не используется, так как сложно построить генератор сигнала синхронизации с такими жесткими требованиями к его длительности.

Обычно используется структура автомата с двойной памятью. Структурная схема такого автомата представлена на рис. 5.54. Блок памяти состоит из двух ступеней памяти, каждая из которых тактируется своим сигналом синхронизации: первая (входная) – сигналом C_1 , вторая (выходная) – сигналом C_2 .

Входная ступень блока памяти служит для запоминания нового состояния автомата, в то время, когда выходная ступень, определяющая состояние автомата, сохраняет код старого состояния.

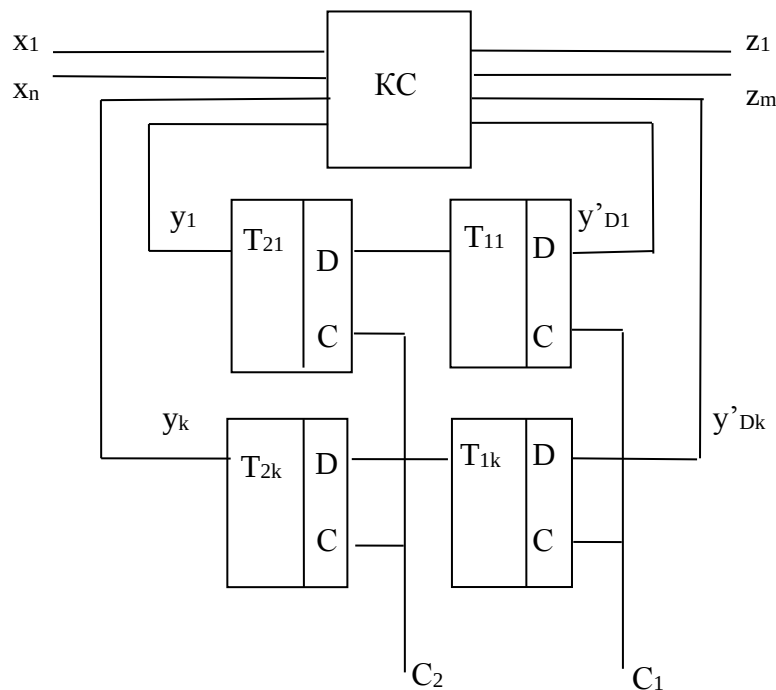


Рис. 5.54

На рис. 5.55 приведена временная диаграмма работы автомата с двойной памятью:

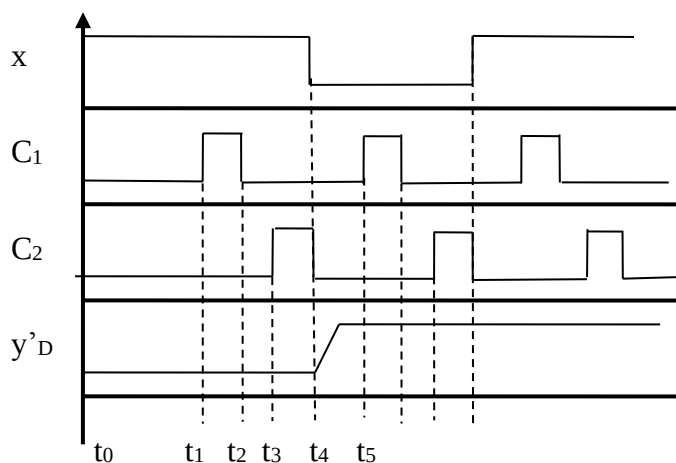


Рис. 5.55

К начальному моменту времени t_0 обе ступени блока памяти должны быть установлены в начальное состояние и на входы автомата поданы нужные значения входных сигналов x_i .

В интервале времени $t_0 - t_1$ в комбинационной схеме автомата происходит формирование значений функций возбуждения и выхода автомата.

К моменту t_1 эти значения должны быть сформированы и по фронту сигнала синхронизации $C_1 = 1$ в течение интервала $t_1 - t_2$ происходит запись нового значения кода состояния в первую (входную) ступень блока памяти, при этом в это время может возникнуть состязание элементов памяти первой ступени, но это состязание не будет влиять на работу автомата, так как запись во вторую ступень запрещена, потому что $C_2 = 0$ и на выходах блока памяти сохраняется код старого состояния. Таким образом, состязание элементов памяти первой ступени является не критическим. Длительность сигнала C_1 должна быть достаточна для записи кода нового состояния во все триггеры первой ступени.

К моменту t_3 состязание элементов памяти первой ступени должны быть закончены и на входах второй ступени блока памяти должны быть установлены значения кода нового состояния. По фронту сигнала синхронизации $C_2 = 1$ в течение интервала $t_3 - t_4$ происходит перезапись значения кода нового состояния из первой (входной) ступени во вторую (выходную) ступень блока памяти. При этом могут возникнуть состязание элементов памяти второй ступени, но это состязание не будет влиять на работу автомата, так как запись возникших неправильных значений функций возбуждения в первую (входную) ступень запрещена, потому что $C_1 = 0$. Таким образом, состязание элементов памяти второй ступени блока памяти также будет не критическим. Длительность сигнала C_2 должна быть достаточна для записи кода нового состояния во все триггеры второй ступени блока памяти.

В интервале $t_3 - t_4$, при $C_2 = 1$ меняются значения на выходах элементов блока памяти y_j , которые поступают на входы комбинационной схемы, следовательно, целесообразно в это же время менять и входные сигналы x_i .

В интервал $t_4 - t_5$ в комбинационной схеме автомата формируются функции выхода и функции возбуждения. Длительность этого интервала выбирается минимально необходимой для формирования функции возбуждения и функции выхода и определяется глубиной комбинационной схемы.

К моменту времени t_5 при $C_1 = 1$ значения на выходах комбинационной схемы уже стабильны, и при $C_1 = 1$ можно считывать значения функций выхода z_j .

Структурная схема автомата с двойной памятью и двумя сериями сигналов синхронизации используется в управляющих устройствах, которые эксплуатируются в неблагоприятных условиях при больших колебаниях температуры внешней среды.

В нормальных условиях обычно вместо двух серий сигналов синхронизации используют прямое и инверсное значение одного сигнала

синхронизации. При $C = 1$ срабатывает первая (входная) ступень, а при $C = 0$ вторая (выходная) ступень блока памяти.

При синхронизации двойной памяти одной серией сигналов синхронизации в качестве элементов памяти используются синхронные триггеры с логической задержкой, т.е. двухступенчатые триггеры, построенные по схеме Master-Slave.

Пример реализации синхронного автомата структурой с двойной памятью и одной серией сигнала синхронизации

Пусть задан граф автомата Мили, рассмотренный ранее в каноническом методе синтеза и реализованный ранее структурой с одной ступенью памяти (рис. 5.56).

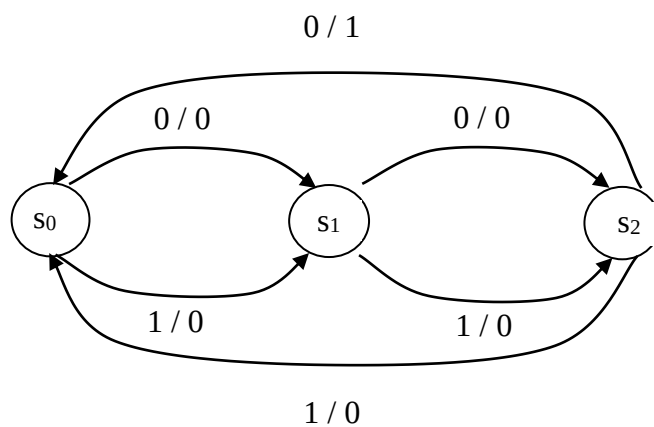


Рис. 5.56

Таблица 5.19

y ₁ y ₂ \ x	x	
	0	1
0 1	11 / 0	11 / 0
1 1	10 / 0	10 / 0
1 0	01 / 1	01 / 0

После произвольного кодирования трех состояний автомата следующим образом: $s_0 = 01$, $s_1 = 11$, $s_2 = 10$ была получена кодированная таблица переходов и выходов (табл. 5.19) и по этой таблице была построена следующая система булевых функций возбуждения элементов памяти и выхода автомата:

$$z = \bar{x} \bar{y}_2;$$

$$y'_{s1} = \bar{y}_1, \quad y'_{R1} = \bar{y}_2;$$

$$y'_{s2} = \bar{y}_2, \quad y'_{R2} = y_1 y_2.$$

Структурная схема синтезируемого автомата с двойной памятью представлена на рис. 5.54, только в качестве элемента памяти используется RS-триггер. Комбинационная схема реализует систему следующих пяти булевых функций: z – функция выхода, y'_{s1} , y'_{R1} – функции возбуждения триггера первой ступени T_{11} , y'_{s2} , y'_{R2} – функции возбуждения триггера первой ступени T_{12} , зависящие от трех переменных: x , y_1 , y_2 .

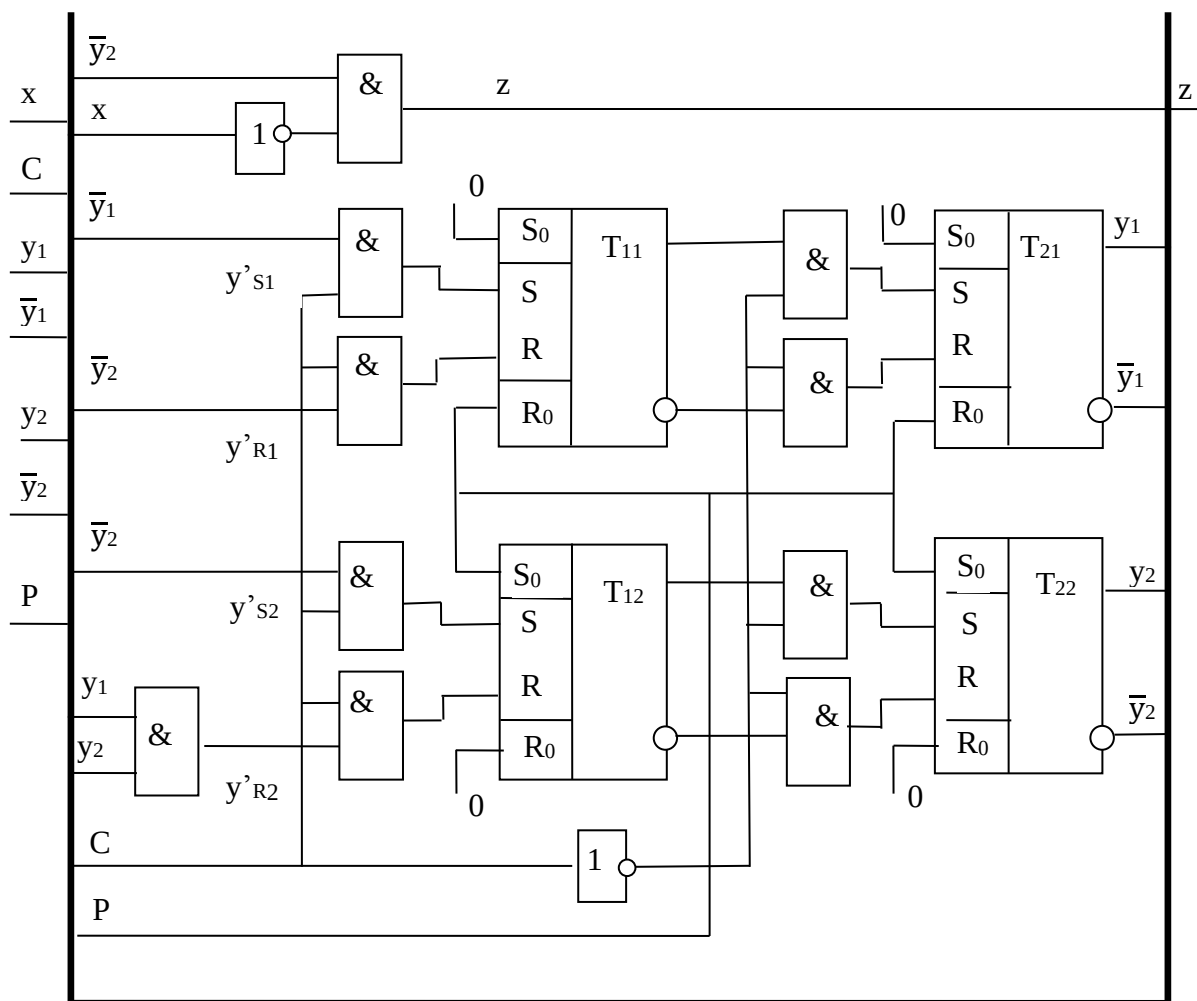


Рис. 5.57

К выходам логических схем, реализующих функции возбуждения, подключаются элементы памяти, образующие двойную память автомата.

В результате получается следующая функциональная схема автомата с двойной памятью (рис. 5.57). Для проверки правильности ее построения необходимо протестировать его работу на конкретном входном слове. На рис. 5.58 представлена временная диаграмма работы автомата по переработке входного слова $1 \rightarrow 1 \rightarrow 0$. Это слово перерабатывается в выходное слово $0 \rightarrow 0 \rightarrow 1$.

По сигналу $P = 1$ к моменту времени t_0 обе ступени блока памяти должны быть установлены в начальное состояние $s_0 = 01$ и на входы автомата поданы нужные значения входного сигнала x .

В интервале времени $t_0 - t_1$ в комбинационной схеме автомата происходит формирование значений функций возбуждения и выхода автомата.

К моменту t_1 эти значения должны быть сформированы и по фронту сигнала синхронизации $C = 1$ в течение интервала $t_1 - t_2$ происходит запись нового значения кода состояния в первую (входную) ступень блока памяти, при этом в это время может возникнуть состязание элементов

памяти первой ступени, но это состязание не будет влиять на работу автомата, так как запись во вторую ступень запрещена, потому что $C = 0$ и на выходах блока памяти сохраняется код старого состояния. Таким образом, состязание элементов памяти первой ступени является некритическим. Длительность сигнала $C = 1$ должна быть достаточна для записи кода нового состояния во все триггеры первой ступени.

К моменту t_2 состязание элементов памяти первой ступени должны быть закончены и на входах второй ступени блока памяти должны быть установлены значения кода нового состояния. По спаду сигнала синхронизации $C = 1$ в течение интервала $t_2 - t_3$ происходит перезапись значения кода нового состояния из первой (входной) ступени во вторую (выходную) ступень блока памяти. При этом могут возникнуть состязание элементов памяти второй ступени, но это состязание не будет влиять на работу автомата, так как запись возникших неправильных значений функций возбуждения в первую (входную) ступень запрещена, потому что $C = 0$. Таким образом, состязание элементов памяти второй ступени блока памяти также будет некритическим. Длительность сигнала C должна быть достаточна для записи кода нового состояния во все триггеры второй ступени блока памяти.

В интервале $t_2 - t_3$, при $C = 0$ меняются значения на выходах элементов блока памяти y_1 и y_2 , которые поступают на входы комбинационной схемы, следовательно, целесообразно в это же время менять и входной сигнал x . Далее в этом же интервале $t_2 - t_3$ в комбинационной схеме автомата формируются значения функции выхода и функции возбуждения. Длительность этого интервала выбирается минимально необходимой для формирования функции возбуждения и функции выхода после изменения значений входных сигналов и определяется глубиной комбинационной схемы.

К моменту времени t_3 при $C = 1$ значения на выходах комбинационной схемы уже стабильны и по фронту $C = 1$ можно считывать значения функции выхода z и значения функции возбуждения элементов памяти. Далее этот процесс повторяется до переработки всего входного слова и возврата автомата в начальное состояние.

Расчет параметров сигнала синхронизации

Длительность сигнала синхронизации выбирается следующим образом: вначале определяются логические элементы, которые меняют свое состояние при этом значении синхросигнала и определяется их число в самой длинной цепочке из этих элементов. Длительность определяется из условия, что все элементы, которые должны изменить свое состояние, успели это сделать.

Тогда t_1 – длительность сигнала синхронизации $C = 1$ определяется временем срабатывания триггеров первой ступени $t_1 = \Delta t + 2 \Delta \tau = 3 \Delta \tau$, где $2\Delta \tau$ – время срабатывания базового RS-триггера. Длительность t_0 – сигнала синхронизации $C = 0$ определяется суммой времён срабатывания триггеров

второй ступени: $2 \Delta\tau + 2 \Delta\tau = 4 \Delta\tau$ и времени формирования в комбинационной схеме значений функций - $\Delta\tau$, так как длина цепочки в комбинационной схеме равна единице, итого: $t_0 = 4 \Delta\tau + \Delta\tau = 5 \Delta\tau$.

$$T = t_1 + t_0 = 3 \Delta\tau + 5 \Delta\tau = 8 \Delta\tau$$

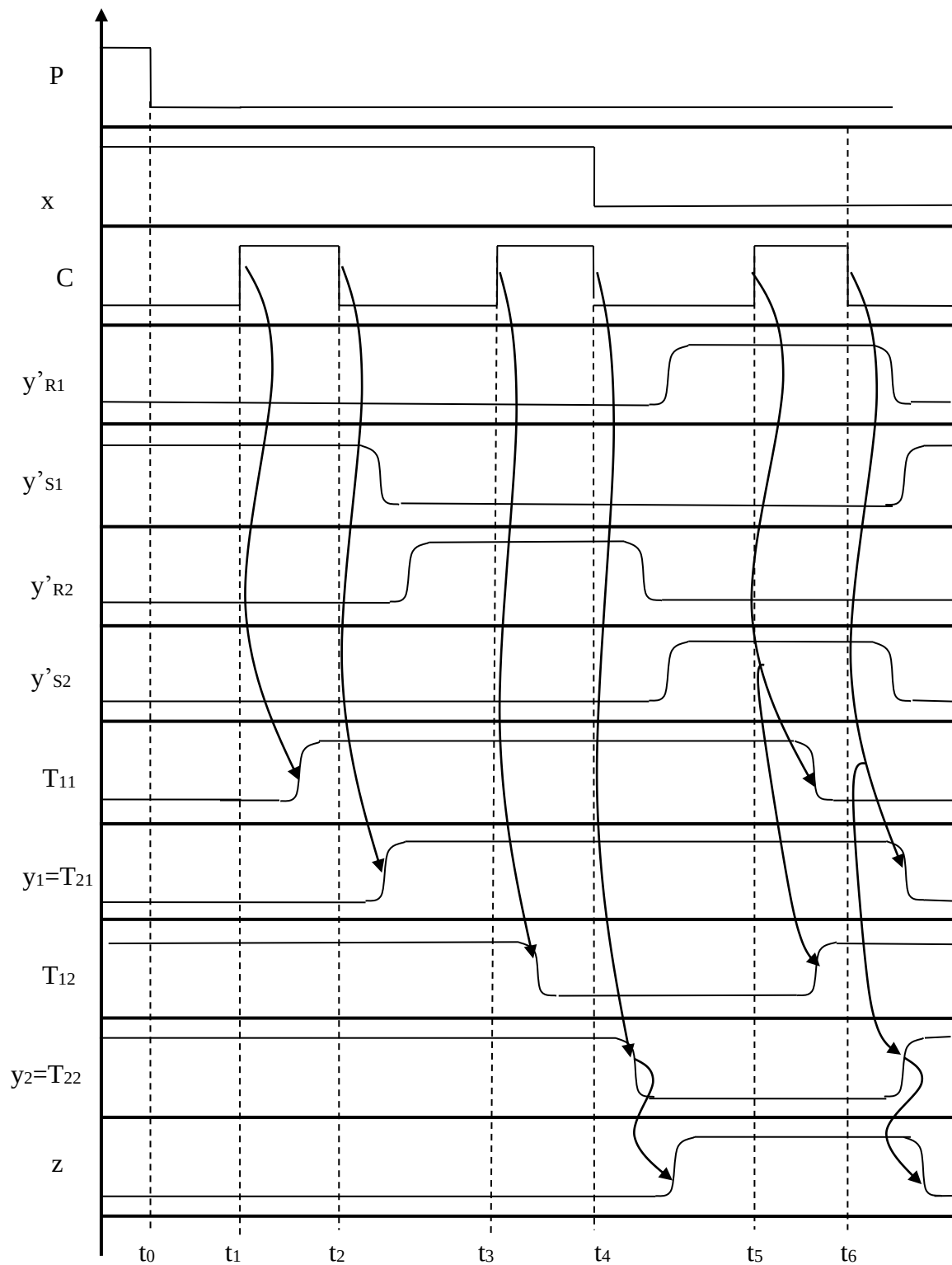


Рис. 5.58

5.6. ПОСТРОЕНИЕ КОМБИНАЦИОННОЙ СХЕМЫ АСИНХРОННОГО АВТОМАТА

Существенным отличием структурного синтеза асинхронного автомата от синтеза синхронного автомата является необходимость построения комбинационной схемы без так называемого “явления риска”. Причина этого явления состоит в том, что реальные логические элементы обладают определенной задержкой во времени срабатывания, причем величина задержки заранее не известна и зависит от технологии изготовления и от внешних факторов в процессе эксплуатации, поэтому определить соотношение задержек различных логических элементов заранее невозможно.

Для анализа этого явления используется следующая модель реального логического элемента, которая состоит из идеального логического элемента (ИЛЭ) и элемента задержки $\Delta\tau$ (рис.5.59). Задержкой обладают также линии связи, но эти задержки относят к задержкам логических элементов.

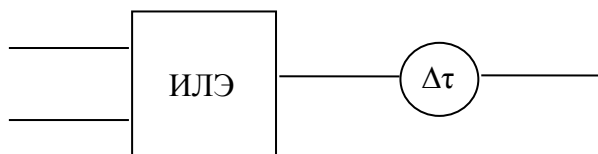


Рис.5.59

Величина задержки $\Delta\tau$ зависит от температуры, от расположения в схеме, нагрузки, влажности и других факторов. Величина задержки по техническому паспорту может меняться в следующих пределах:

$$\Delta\tau_{\min} < \Delta\tau < \Delta\tau_{\max}$$

Из-за разброса значений $\Delta\tau$ разных логических элементов в схеме из этих элементов возникает так называемое явление риска (далее просто риск). Риск возникает при смене одного набора значений входных переменных на другой набор значений и проявляется в появлении на выходе схемы кратковременного неправильного значения реализуемой функции.

На рис. 5.60 приведена логическая схема, реализующая функцию трех переменных: $f(x_1x_2x_3) = \bar{x}_1x_3 \vee x_2\bar{x}_3$:

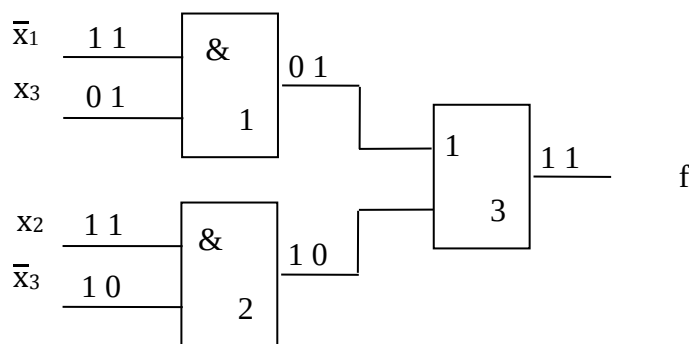


Рис. 5.60

Пусть существует два набора значений входных переменных $x_1x_2x_3$: $\sigma_1 = 010$ и $\sigma_2 = 011$, таких что $f(\sigma_1) = f(\sigma_2) = 1$. Пусть задержка первого элемента Δ_1 больше задержки второго элемента Δ_2 , т.е. $\Delta_1 > \Delta_2$ и пусть для простоты $\Delta_3 = 0$.

Временная диаграмма работы схемы при смене набора σ_1 на набор σ_2 приведена на рис. 5.61.

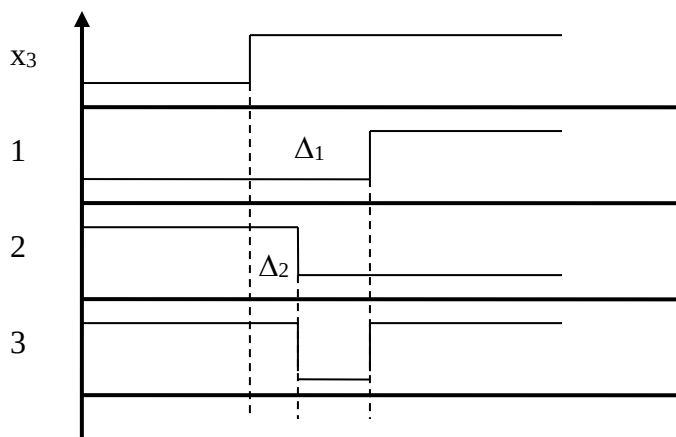


Рис. 5.61

На рис. 5.61 видно, что на выходе схемы ненадолго появляется 0 - это называется явлением риска.

Статический риск.

Определение: пусть существует функция $f(\vec{x})$ и существуют два набора значений входных переменных σ_1 и σ_2 , таких что $f(\sigma_1) = f(\sigma_2)$, тогда, если при смене набора σ_1 на σ_2 на выходе схемы появляется значение, отличное от $f(\delta_1)$, то говорят что в схеме имеет место *статический риск*.

Статические риски бывают:

а) статический риск в «1» (рис. 5.62).

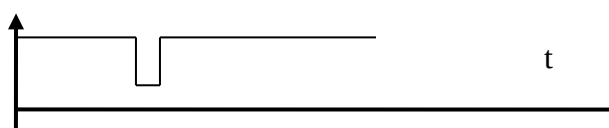


Рис. 5.62

б) статический риск в «0» (рис. 5.63).

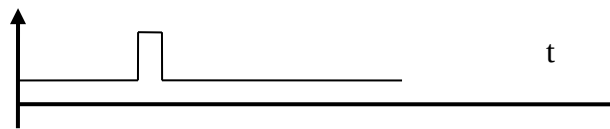


Рис. 5.63

Следует отметить, что если $\Delta_1 < \Delta_2$, то статический риск в «1» в схеме (рис. 5.61) отсутствует.

Динамический риск.

Определение: пусть существует функция $f(\tilde{x})$ и существуют два набора значений входных переменных σ_1 и σ_2 , таких, что $f(\sigma_1) \neq f(\sigma_2)$, тогда, если при смене набора σ_1 на σ_2 на выходе схемы появляется следующая последовательность значений: $f(\sigma_1) \rightarrow f(\sigma_2) \rightarrow f(\sigma_1) \rightarrow f(\sigma_2)$, то говорят, что в схеме имеет место *динамический риск*.

На рис. 5.64 приведена логическая схема, реализующая функцию пяти переменных: $f(\tilde{x}) = (\bar{x}_1 x_2 \vee x_1 x_3)(\bar{x}_1 x_4 \vee x_5)$:

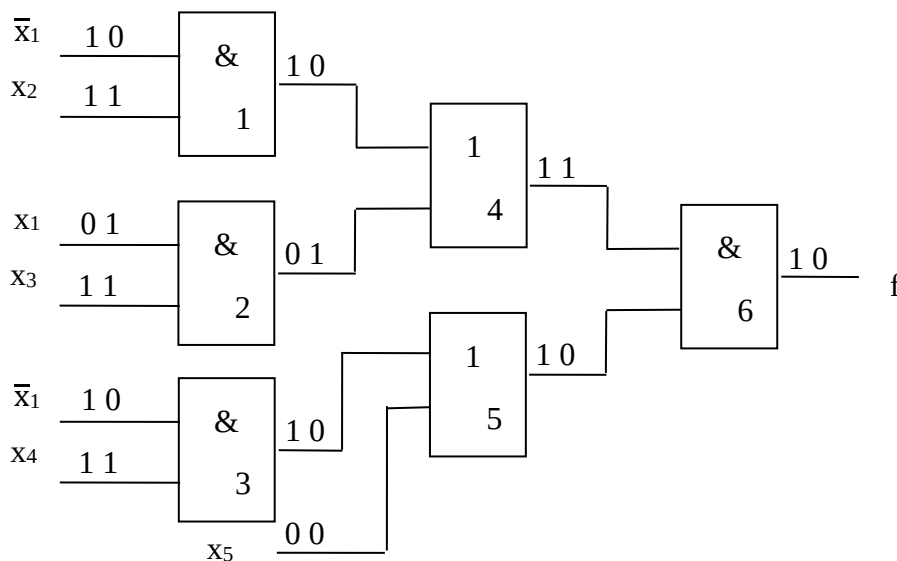


Рис. 5.64

Пусть существует два набора значений переменных $x_1 x_2 x_3 x_4 x_5$: $\sigma_1 = 01110$ и $\sigma_2 = 11110$, таких, что $f(\sigma_1) = 1$, $f(\sigma_2) = 0$. Пусть между задержками первых трех элементов существуют следующие соотношения: $\Delta_1 < \Delta_2 < \Delta_3$ и пусть для простоты $\Delta_4 = \Delta_5 = \Delta_6 = 0$.

Временная диаграмма работы схемы при смене набора σ_1 на набор σ_2 приведена на рис. 5.65.

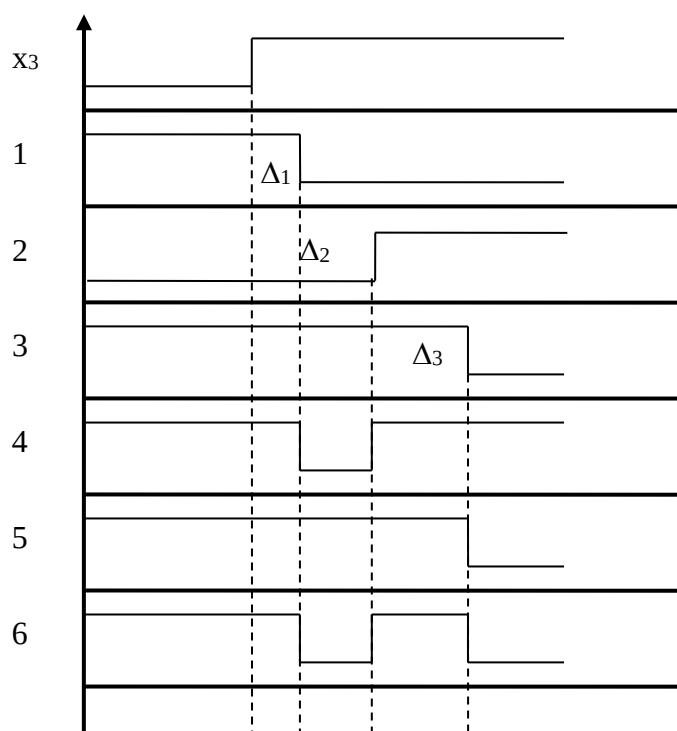


Рис. 5.65

Из временной диаграммы видно, что при таком соотношении между задержками логических элементов в схеме имеет место динамический риск.

Существует динамический риск при переключении $0 \rightarrow 1$ и динамический риск $1 \rightarrow 0$ (рис. 5.66).

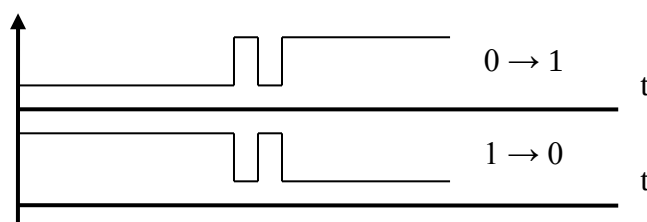


Рис. 5.66

Наличие риска в комбинационной схеме автомата приводит к его неправильной работе.

В синхронном автомате явление риска исключается за счет выбора периода и длительности сигнала синхронизации. Длительность синхросигнала должен быть больше времени срабатывания самой длинной цепочки элементов в схеме. За это время все процессы по формированию значений функций должны быть завершены.

Для построения асинхронного автомата необходимо использовать комбинационные схемы без риска.

Построение схем без риска.

В рассматриваемом способе исключения явления риска на комбинационные схемы накладываются следующие ограничения:

- при смене набора значений входных переменных не может одновременно меняться значение более чем одной переменной;

- глубина комбинационной схемы не должна быть более двух, т.е. схема должна иметь не более двух ярусов.

Наличие этих двух ограничений означает, что использование этого способа позволяет устранить только статический риск.

В двухъярусной схеме возможно два варианта схем:

- 1) схема построенная по дизъюнктивной нормальной форме (ДНФ).
 $f_{\text{ДНФ}} = k_1 \vee k_2 \vee \dots \vee k_m$, где k_i – конъюнкция (рис. 5.67);
- 2) схема построенная по конъюнктивной нормальной форме (КНФ).
 $f_{\text{КНФ}} = d_1 \wedge d_2 \wedge \dots \wedge d_n$, где d_i – дизъюнкция (рис. 5.68).

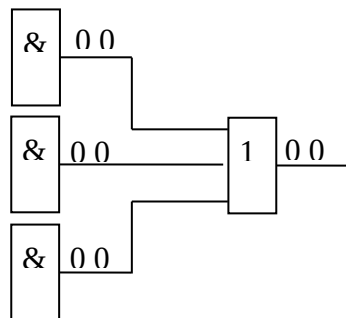


Рис. 5.67

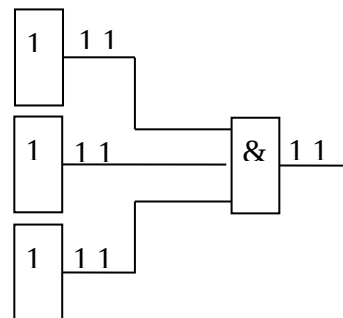


Рис. 5.68

Из анализа этих схем можно сделать следующие утверждения:

- 1) в двухъярусной схеме, построенной по ДНФ, статический риск в «0» отсутствует;
- 2) в двухъярусной схеме, построенной по КНФ статический риск в «1» отсутствует;
- 3) пусть есть схема, построенная по ДНФ и пусть переменная x_i входит в эту ДНФ только в прямом или только инверсном виде, тогда при изменении значения этой переменной статический риск в схеме отсутствует;
- 4) если есть схема, построенная по КНФ и если переменная x_i входит в эту КНФ только в прямом или только инверсном виде, тогда при изменении значения этой переменной статический риск в схеме отсутствует;
- 5) если есть схема, построенная по ДНФ, и если переменная x_i входит в эту ДНФ как в прямом, так и в инверсном виде, тогда в схеме может возникнуть статический риск в «1» при изменении значения этой переменной.

Для доказательства исходная ДНФ представляется в виде: $f_{\text{ДНФ}} = A \vee B(x_i) \vee C(\bar{x}_i)$, где A – группа конъюнкций, не содержащих x_i ; $B(x_i)$ – группа конъюнкций, содержащих x_i в прямом виде; $C(\bar{x}_i)$ – группа конъюнкций, содержащих x_i в инверсном виде (рис. 5.69).

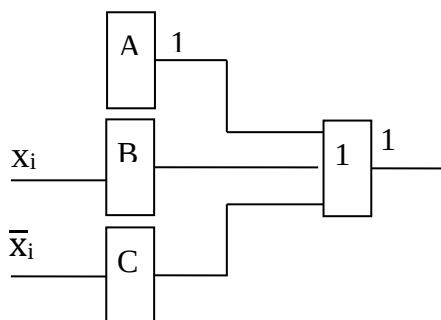


Рис. 5.69

Пусть σ_1 и σ_2 два набора значений переменных, отличающихся значением только по переменной x_i , причем $f(\sigma_1) = f(\sigma_2) = 1$.

Возможны различные ситуации:

Если $A(\sigma_1) = A(\sigma_2) = 1$, то в этом случае риска нет, так как значение «1» на выходе группы A обеспечит значение «1» на выходе всей схемы.

Если $A(\sigma_1) = A(\sigma_2) = 0$, то в этом случае значение «1» на выходе всей схемы обеспечивается конъюнкциями группы B или конъюнкциями группы C, т.е. возможны два варианта:

1. $B(\sigma_1) = 1$ $C(\sigma_1) = 0$;

$B(\sigma_2) = 0$, $C(\sigma_2) = 1$.

При этом возможны следующие ситуации:

а) $\Delta_B = \Delta_C$ – риска нет, но это нереальная ситуация;

б) $\Delta_B < \Delta_C$ – риск есть (рис. 5.70);

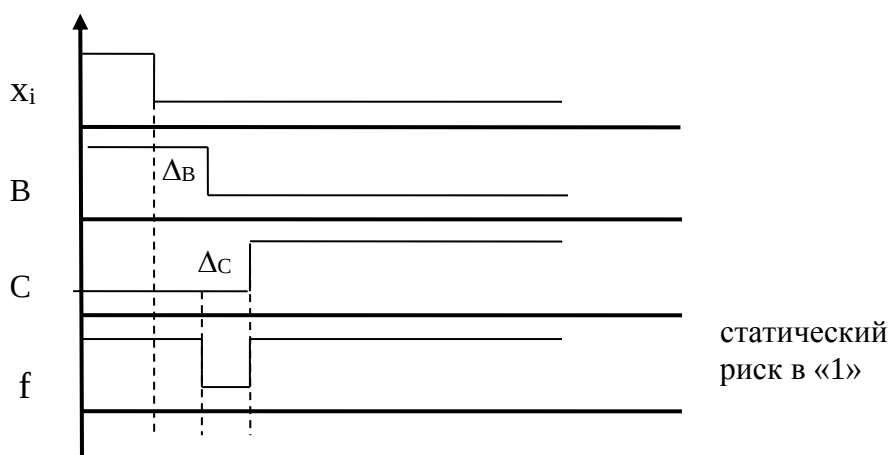


Рис. 5.70

в) $\Delta_B > \Delta_C$ – риска нет (рис. 5.71)

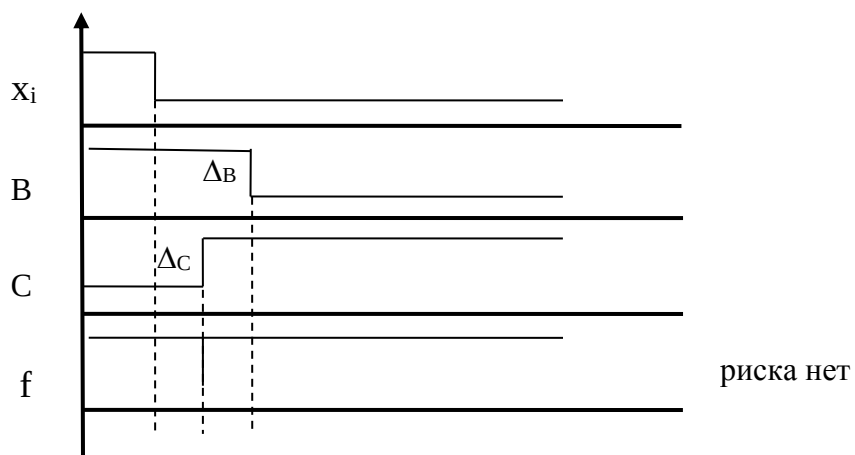


Рис. 5.71

2. $B(\sigma_1) = 0$ $C(\sigma_1) = 1$;
 $B(\sigma_2) = 1$, $C(\sigma_2) = 0$.

При этом возможны следующие ситуации:

- а) $\Delta_B = \Delta_C$ – риска нет, но это нереальная ситуация;
б) $\Delta_B < \Delta_C$ – риска нет (рис. 5.72);

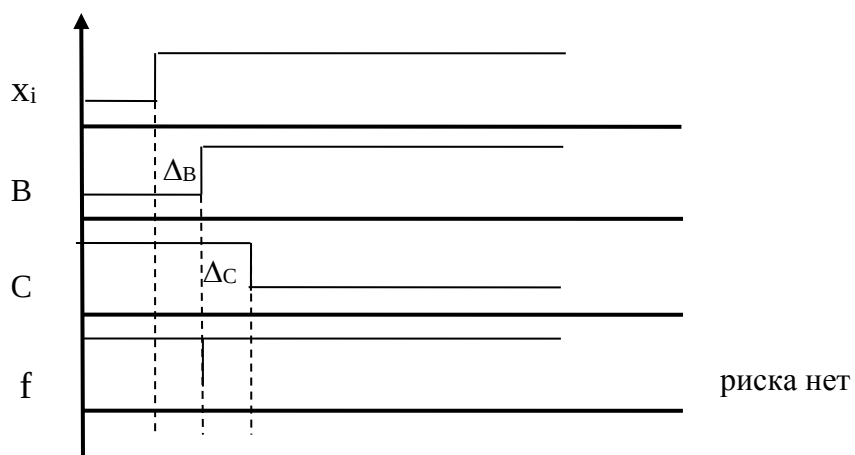


Рис. 5.72

- в) $\Delta_B > \Delta_C$ – риск есть (рис. 5.73).

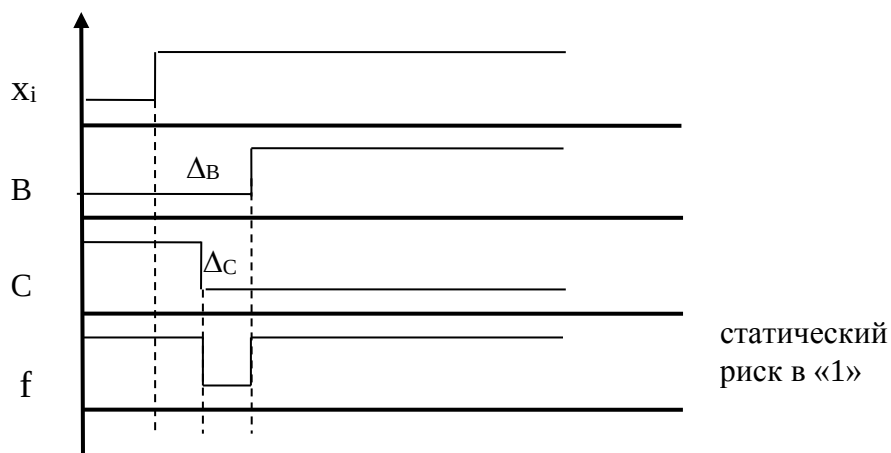


Рис. 5.73

- 6) если есть схема, построенная по КНФ, и если переменная x_i входит в эту КНФ, как в прямом, так и в инверсном виде, тогда в схеме может возникнуть статический риск в «0» при изменении значения этой переменной.

Для доказательства исходная КНФ представляется в виде:

$f_{\text{КНФ}} = A \wedge B(x_i) \wedge C(\bar{x}_i)$, где A – группа дизъюнкций, не содержащих x_i ; $B(x_i)$ – группа дизъюнкций, содержащих x_i в прямом виде; $C(\bar{x}_i)$ – группа дизъюнкций, содержащих x_i в инверсном виде (рис. 5.74).

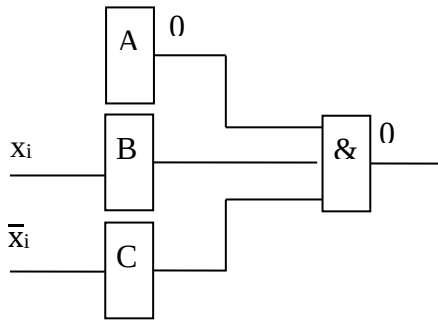


Рис. 5.74

Пусть σ_1 и σ_2 два набора значений переменных отличающихся значением только по переменной x_i , причем $f(\sigma_1) = f(\sigma_2) = 0$.

Возможны различные ситуации:

$A(\sigma_1) = A(\sigma_2) = 0$ – то в этом случае риска нет, так как «0» на выходе группы A обеспечит значение «0» на выходе всей схемы.

$A(\sigma_1) = A(\sigma_2) = 1$ – то в этом случае значение «0» на выходе всей схемы обеспечивается дизъюнкциями группы B или дизъюнкциями группы C , т.е. возможны два варианта:

1. $B(\sigma_i) = 0$ $C(\sigma_i) = 1$;
 $B(\sigma_i) = 1$ $C(\sigma_i) = 0$.

При этом возможны следующие ситуации:

- а) $\Delta_B = \Delta_C$ – риска нет, но это нереальная ситуация;
- б) $\Delta_B < \Delta_C$ – риск есть (рис. 5.75);

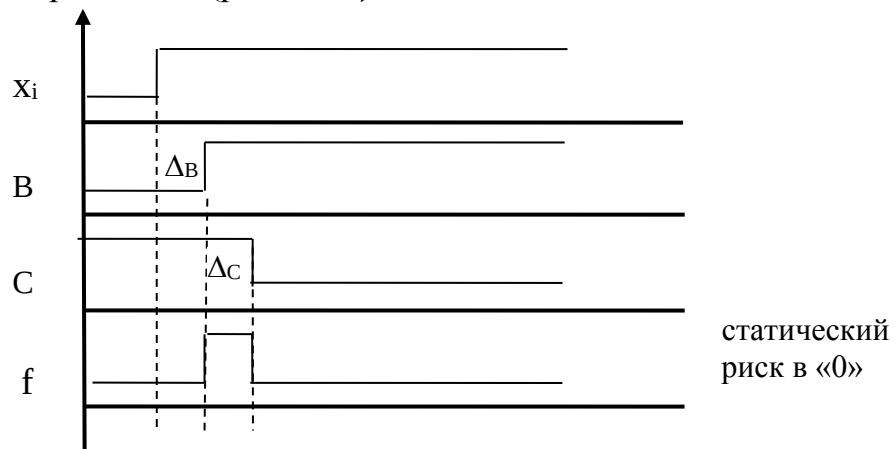


Рис. 5.75

- в) $\Delta_B > \Delta_C$ – риска нет (рис. 5.76).

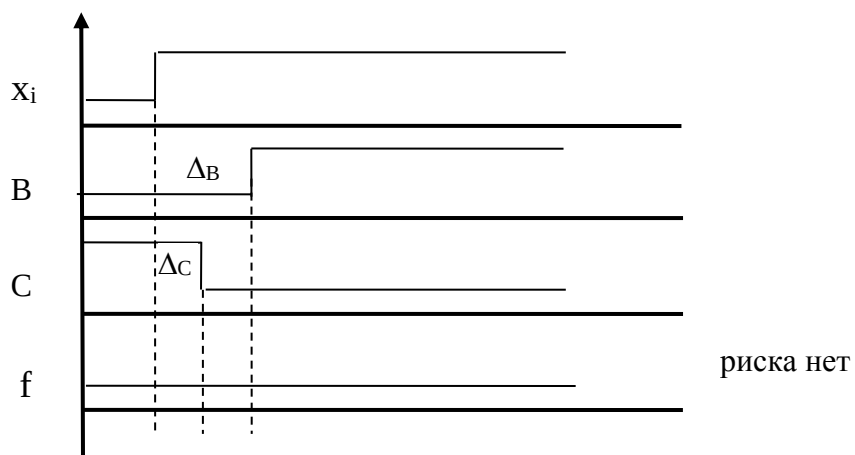


Рис. 5.76

2. $B(\sigma_i) = 1$ $C(\sigma_i) = 0$;

$B(\sigma_i) = 0$ $C(\sigma_i) = 1$.

а) $\Delta_B = \Delta_C$ – риска нет, но это нереальная ситуация;

б) $\Delta_B < \Delta_C$ – риска нет (рис. 5.77);

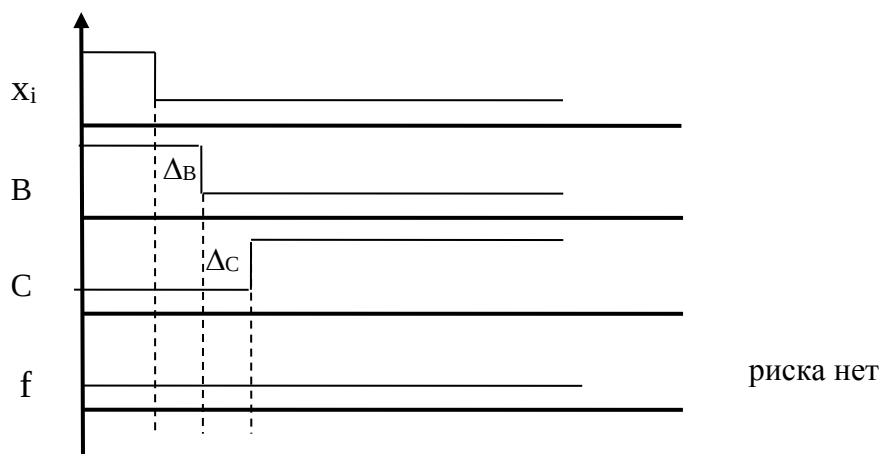


Рис. 5.77

в) $\Delta_B > \Delta_C$ – риск есть (рис. 5.78).

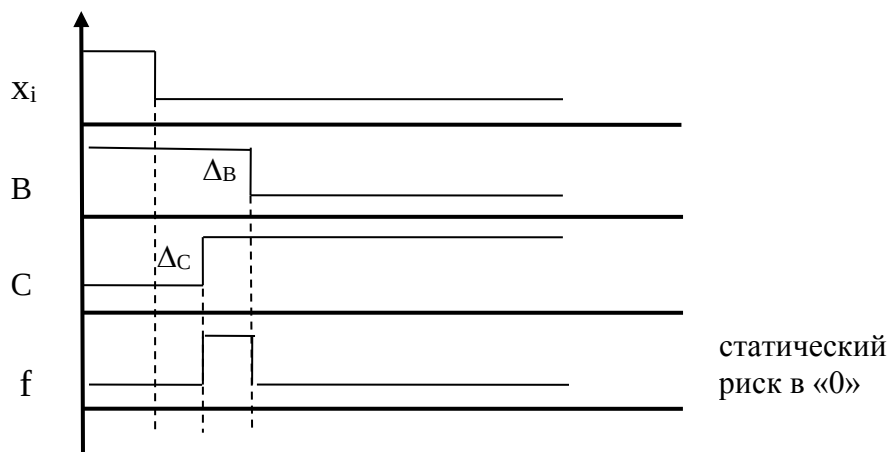


Рис. 5.78

Из приведенных выше утверждений 1) – 6) видно, что в некоторых случаях риск, который может возникнуть из за наличия разных значений на выходах элементов группы В и С, “шунтируется” наличием нужного значения на выходах элементов группы А. Поэтому идея построения схемы без риска заключается в выявлении наборов значений переменных σ_1 и σ_2 , при смене которых может возникнуть риск, и добавлении к схеме элемента группы А, который обеспечит отсутствие риска в схеме для этой конкретной пары наборов σ_1 и σ_2 .

Алгоритм построения схемы без риска по ДНФ.

Пусть задана в ДНФ функция $f(x_1x_2\dots x_n)$ от n переменных.

- 1) задается $i := 1$ – индекс переменной x_i ;
- 2) относительно текущей переменной x_i исходная ДНФ представляется в виде $f_{\text{ДНФ}} = A \vee B(x_i) \vee C(\bar{x}_i)$;
- 3) для x_i определяются пары наборов σ_1 и σ_2 соседних по переменной x_i и для которых $f(\sigma_1) = f(\sigma_2) = 1$;
- 4) для каждой такой пары σ_1 и σ_2 определяются значения конъюнкций группы А.
- 5) Если $A(\sigma_1) = A(\sigma_2) = 0$, то риск возможен, и для выявленной пары σ_1 и σ_2 образуется конъюнкция, которая сохраняет значение 1 на обоих наборах σ_1 и σ_2 и не содержит x_i . Полученная конъюнкция добавляется к ДНФ, а в схему включается конъюнктор, реализующий эту конъюнкцию;
- 6) индекс переменной увеличивается; $i := i + 1$ и, если $i \leq n$, то осуществляется переход к п.2, иначе конец работы алгоритма.

Пример построения схемы без риска по ДНФ:

Пусть $f_{\text{ДНФ}}(x_1x_2x_3) = \bar{x}_1\bar{x}_3 \vee x_2x_3$.

Риск возможен по переменной x_3 , так как только она входит в ДНФ как в прямом, так и в инверсном виде. $B = x_2x_3$, $C = \bar{x}_1\bar{x}_3$.

Согласно алгоритму определяются наборы: $\sigma_1 = 010$, $\sigma_2 = 011$ и для этой пары наборов образуется конъюнкция группы А = $\bar{x}_1x_2$.

Окончательно исходная ДНФ примет вид: $f_{\text{ДНФ}}(x_1x_2x_3) = \bar{x}_1\bar{x}_3 \vee x_2x_3 \vee \bar{x}_1x_2$. Реализующая эту ДНФ схема приведена на рис. 5.79.

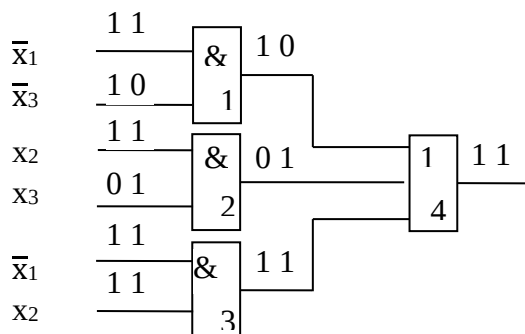


Рис. 5.79

На рис. 5.80 показана временная диаграмма работы схемы для случая $\Delta_1 < \Delta_2$ и с конъюнктом 3, добавленным для исключения статического риска в «1», обозначенного пунктиром.

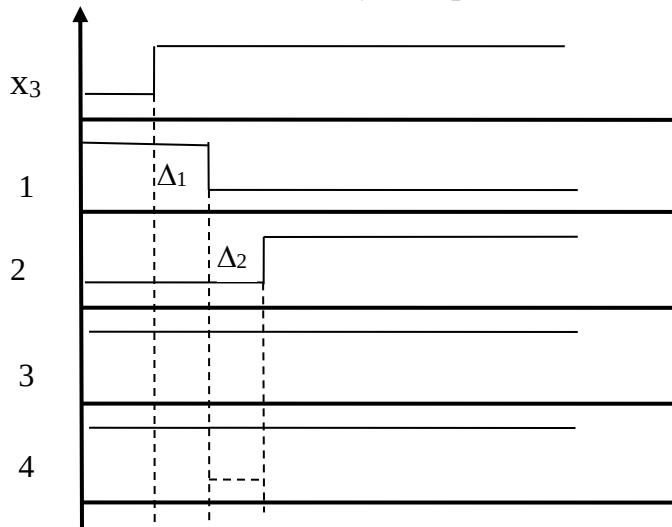


Рис. 5.80

Существует два способа для определения конъюнкций, исключающих явление риска:

- с помощью операции обобщенного склеивания;
 $R x_i \vee S \bar{x}_i = R S \vee R x_i \vee S \bar{x}_i$, где R, S – произвольные конъюнкции.
 Для рассмотренного примера $f_{\text{ДНФ}}(x_1 x_2 x_3) = \bar{x}_1 \bar{x}_3 \vee x_2 x_3$, $R = x_2$, $S = \bar{x}_1$ и следовательно $RS = x_2 \bar{x}_1$;
- с помощью карт Карно.

В карте Карно соседним клеткам соответствуют наборы, отличающиеся значением только в одном разряде x_i , и если эти соседние клетки входят в разные покрытия, то им соответствуют двоичные наборы, при смене которых, на входе схемы возможен риск по переменной x_i . Для исключения этого риска надо построить дополнительное покрытие для этих клеток, которое будет определять конъюнкцию, не содержащую x_i (рис. 5.81). Для примера $f_{\text{ДНФ}}(x_1 x_2 x_3) = \bar{x}_1 \bar{x}_3 \vee x_2 x_3$ первой конъюнкции соответствует верхнее левое покрытие, а второй – вертикальное покрытие. Искомые наборы $\sigma_1 = 010$ и $\sigma_2 = 011$ соответствуют соседним клеткам, входящим в различные покрытия. Дополнительное покрытие этих клеток определяет конъюнкцию $\bar{x}_1 x_2$, исключающую риск по переменной x_3 .

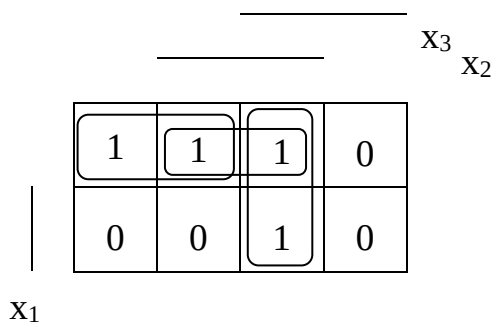


Рис. 5.81

Алгоритм построения схемы без риска по КНФ.

Пусть задана в КНФ функция $f(x_1x_2\dots x_n)$ от n переменных.

- 1) задается $i := 1$ – индекс переменной x_i ;
- 2) относительно текущей переменной x_i исходная КНФ представляется в виде: $A \wedge B(x_i) \wedge C(\bar{x}_i)$;
- 3) для x_i определяются пары наборов σ_1 и σ_2 соседних по переменной x_i и для которых $f(\sigma_1) = f(\sigma_2) = 0$;
- 4) для каждой такой пары σ_1 и σ_2 определяются значения дизъюнкций группы A ;
- 5) Если $A(\sigma_1) = A(\sigma_2) = 1$, то риск возможен, и для выявленной пары σ_1 и σ_2 образуется дизъюнкция, которая сохраняет значение 0 на обоих наборах σ_1 и σ_2 и не содержит x_i . Полученная дизъюнкция добавляется к КНФ, а в схему включается дизъюнктор, реализующий эту дизъюнкцию;
- 6) индекс переменной увеличивается $i := i + 1$ и, если $i \leq n$, то осуществляется переход к п.2, иначе конец работы алгоритма.

Пример построения схемы без риска по КНФ:

Пусть $f_{\text{КНФ}}(x_1x_2x_3) = (\bar{x}_1 \vee x_3) \wedge (x_2 \vee \bar{x}_3)$.

Риск возможен по переменной x_3 , так как только она входит в КНФ, как в прямом, так и в инверсном виде. $B = \bar{x}_1 \vee x_3$, $C = x_2 \vee \bar{x}_3$.

Согласно алгоритму определяются наборы: $\sigma_1 = 100$, $\sigma_2 = 101$ и для этой пары наборов образуется дизъюнкция группы $A = \bar{x}_1 \vee x_2$.

Окончательно исходная КНФ примет вид:

$f_{\text{КНФ}}(x_1x_2x_3) = (\bar{x}_1 \vee x_3) \wedge (x_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee x_2)$. Реализующая эту КНФ схема приведена на рис. 5.82.

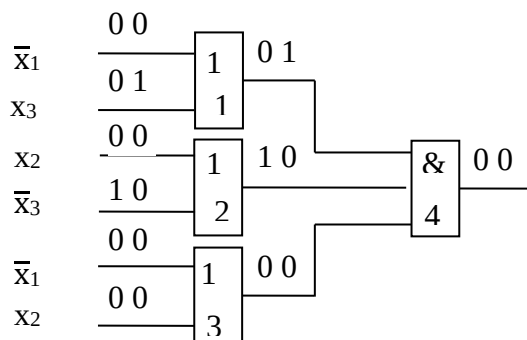


Рис. 5.82

На рис. 5.83 показана временная диаграмма работы схемы для случая $\Delta_1 < \Delta_2$ и с дизъюнктором 3, добавленным для исключения статического риска в «0» и обозначенного пунктиром.

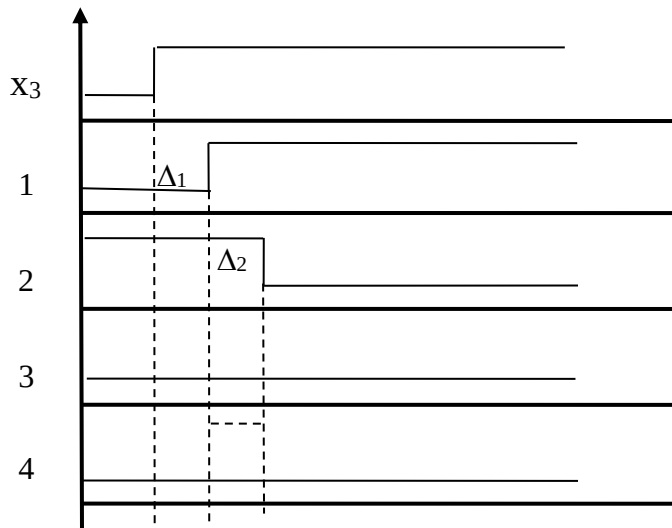


Рис. 5.83

Для определения дизъюнкций, исключающих явление риска в схеме, построенной по КНФ целесообразно от функции заданной в КНФ, перейти к заданной в ДНФ её инверсной функции.

Для приведенного примера $f_{\text{КНФ}}(x_1 x_2 x_3) = (\bar{x}_1 \vee x_3) \wedge (x_2 \vee \bar{x}_3)$ её инверсная функция $\bar{f}_{\text{ДНФ}}(x_1 x_2 x_3) = x_1 \bar{x}_3 \vee \bar{x}_2 x_3$. С помощью операции обобщенного склеивания получается: $\bar{f}_{\text{ДНФ}}(x_1 x_2 x_3) = x_1 \bar{x}_3 \vee \bar{x}_2 x_3 \vee x_1 \bar{x}_2$.

На рис. 5.84 приведена карта Карно для инверсной функции:

$\bar{f}_{\text{ДНФ}}(x_1 x_2 x_3) = x_1 \bar{x}_3 \vee \bar{x}_2 x_3$. Первой конъюнкции соответствует нижнее левое покрытие, а второй – вертикальное покрытие. Искомые наборы $\sigma_1 = 100$ и $\sigma_2 = 101$ соответствуют соседним клеткам, входящим в различные покрытия. Дополнительное покрытие этих клеток определяет конъюнкцию $x_1 \bar{x}_2$, исключающую риск по переменной x_3 .

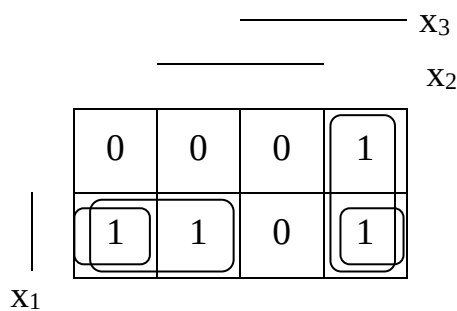


Рис. 5.84

Окончательно ДНФ инверсной функции: $\bar{f}_{\text{ДНФ}}(x_1 x_2 x_3) = x_1 \bar{x}_3 \vee \bar{x}_2 x_3 \vee x_1 \bar{x}_2$.

После перехода к КНФ прямой функции получается тот же результат:

$$f_{\text{КНФ}}(x_1 x_2 x_3) = (\bar{x}_1 \vee x_3) \wedge (x_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee x_2).$$

6. АВТОМАТЫ, ФОРМАЛЬНЫЕ ЯЗЫКИ И ГРАММАТИКИ

6.1. ОСНОВНЫЕ ОПРЕДЕЛЕНИЯ

Язык – это совокупность правильно построенных конструкций (предложений).

Грамматика – это совокупность объектов $\Gamma = \langle V_T, V_N, I, R \rangle$:

V_T – словарь терминальных символов, состоящий из основных символов языка, к которым относятся буквы, цифры, знаки и другие символы языка. Терминальные символы обозначаются строчными буквами латинского алфавита: $V_T = \{a, b, \dots, z\}$;

V_N – словарь нетерминальных символов (вспомогательный), используемый для обозначения частей конструкций языка при записи правил грамматики. Символы нетерминального словаря обозначаются прописными буквами латинского алфавита: $V_N = \{A, B, \dots, Z\}$;

I – начальный символ грамматики, является элементом нетерминального словаря: $I \in V_N$;

R – множество порождающих правил грамматики вида: $\phi \rightarrow \psi$, где ϕ и ψ – некоторые последовательности символов полного словаря грамматики: $V = V_T \cup V_N$, причем, ϕ содержит хотя бы один нетерминальный символ.

Пусть V_T^* – множество конечных цепочек, построенных из символов терминального словаря и V^* – множество цепочек, построенных из символов полного словаря.

Конечные цепочки, построенные из символов полного словаря, обозначаются строчными буквами греческого алфавита $\alpha, \beta, \phi, \psi \in V^*$. Порождающее правило вида: $\phi \rightarrow \psi$ определяет подстановку, при которой последовательность ϕ заменяется на последовательность ψ .

Пусть есть правило $R : \phi \rightarrow \psi$, тогда, если задана цепочка $\omega = \xi_1 \phi \xi_2$, где $\xi_1 \xi_2 \in V^*$ – цепочки и если из неё получается путем применения заданного правила цепочка $\eta = \xi_1 \psi \xi_2$, то говорят, что η непосредственно выводима из ω и обозначается $\omega \Rightarrow \eta$.

Пусть задано множество цепочек:

$\Omega = \{ \omega_1, \omega_2, \omega_3, \dots, \omega_n \}$ таких, что $\omega_1 \Rightarrow \omega_2, \omega_2 \Rightarrow \omega_3, \dots, \omega_{n-1} \Rightarrow \omega_n$, то это обозначается $\omega_1 \Rightarrow^* \omega_n$ и говорят, что ω_n выводима из ω_1 .

Множество конечных цепочек, которые выводятся из начального символа грамматики I и которые представлены только терминальными символами, называются языком, порождаемым грамматикой Γ : $L(\Gamma) = \{ \omega : \omega \in V_T^* \text{ \& } I \Rightarrow^* \omega \}$.

Примеры грамматик и порождаемых ими языков:

$\Gamma_1: V_T = \{a, b, c\}, V_N = \{I\}, I, R = \{I \rightarrow abc\}.$

$L(\Gamma_1) = \{abc\}.$

$\Gamma_2: V_T = \{a, d, c\}, V_N = \{I, B, C\}, I,$

$R = \{I \xrightarrow{1} aB, B \xrightarrow{2} Cd, B \xrightarrow{3} dC, C \xrightarrow{4} c\}$

$$I \xrightarrow{1} aB \xrightarrow{2} aCd \xrightarrow{4} acd.$$

$$I \xrightarrow{1} aB \xrightarrow{3} adC \xrightarrow{4} adc.$$

$$L(\Gamma_2) = \{acd, adc\}$$

$$\Gamma_3: V_T = \{a, b\}, V_N = \{I, A\}, I,$$

$$R = \{I \xrightarrow{1} aAb, aA \xrightarrow{2} aaAb, A \xrightarrow{3} \Lambda\}, \Lambda - \text{пустой символ.}$$

$$I \xrightarrow{1} aAb \xrightarrow{2} aaAbb \xrightarrow{3} aabb.$$

$$I \xrightarrow{1} aAb \xrightarrow{2} aaAbb \xrightarrow{2} aaaAbbb \xrightarrow{3} aaabbb.$$

Последовательность номеров правил, использованных для порождения цепочки, называется *синтаксическим разбором цепочки*. Для последней цепочки: [1, 2, 2, 3].

$$L(\Gamma_3) = \{(a^n b^n), n \geq 1\} - \text{общее выражение для всех цепочек.}$$

$$\Gamma_4: V_T = \{a, b\}, V_N = \{I, A\}, I, R = \{I \rightarrow aA, A \rightarrow bA\}.$$

$$I \xrightarrow{1} aA \xrightarrow{2} abA \xrightarrow{2} abbA \xrightarrow{2} \dots$$

$$L(\Gamma_4) = \emptyset.$$

Грамматика не порождает цепочек, содержащих только терминальные символы, следовательно, это пустой язык.

Утверждение:

Для того чтобы язык $L(\Gamma)$ не был пуст в нем должно быть хотя бы одно правило вида:

$$\eta \rightarrow \omega, \omega \in V_T^* \text{ и чтобы } I \xrightarrow{*} \eta.$$

$$\Gamma_5: V_T = \{a, b\}, V_N = \{I, A\}, I, .$$

$$R = \{I \xrightarrow{1} aAb, A \xrightarrow{2} aAb, A \xrightarrow{3} ab, A \xrightarrow{4} \Lambda\}$$

Синтаксический разбор цепочки может осуществляться в виде синтаксического дерева (рис. 6.1).

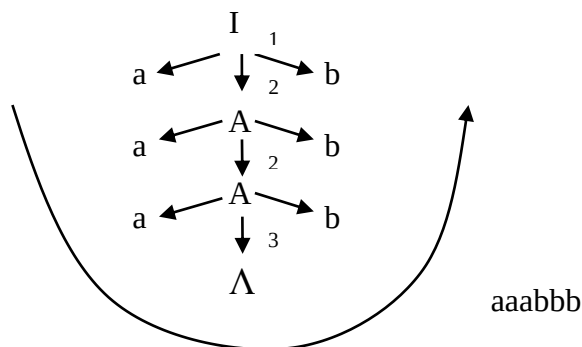


Рис. 6.1

Порожденная цепочка представляет собой конечные вершины дерева, которые выписываются при обходе вершин дерева слева направо против часовой стрелки.

$$L(\Gamma_5) = \{(a^n b^n), n \geq 1\}.$$

Две грамматики называются *эквивалентными*, если они порождают один и тот же язык. Грамматики Γ_3 и Γ_5 – эквивалентны.

Цепочка, порожденная грамматикой, называется *неоднозначной*, если она может быть выведена из начального символа грамматики I более чем одним способом, т.е. цепочка имеет несколько синтаксических разборов.

Грамматика, порождающая неоднозначную цепочку, называется неоднозначной.

$$\begin{aligned} \Gamma_6: \quad V_T &= \{a, b, c\}, V_N = \{I, A, B\}, I, \\ R &= \{I \xrightarrow{1} AB, A \xrightarrow{2} a, A \xrightarrow{3} ac, B \xrightarrow{4} b, B \xrightarrow{5} cb\} \\ I &=^1> AB =^2> aB =^4> ab. \\ I &=^1> AB =^3> acB =^4> acb, \text{ т.е. } [1, 3, 4]. \\ I &=^1> AB =^2> cB =^5> acb, \text{ т.е. } [1, 2, 5]. \end{aligned}$$

Цепочка acb неоднозначна, следовательно, Γ_6 неоднозначна.

Задача распознавания цепочек языка

Задачей распознавания является определение, является ли цепочка порождением заданной грамматики.

Пусть задано правило: $R = \phi \rightarrow \psi$, где $\phi, \psi \in V^*$ и две цепочки

$\omega = \xi_1 \phi \xi_2$ и $\eta = \xi_1 \psi \xi_2$, где $\xi_1 \xi_2 \in V^*$, то говорят, что цепочка ω получается из цепочки η непосредственно сворачиванием и это обозначается: $\omega \leq \eta$.

Пусть задано множество цепочек:

$\Omega = \{\omega_1, \omega_2, \omega_3, \dots, \omega_n\}$ таких, что $\omega_{n-1} \leq \omega_n, \omega_2 \leq \omega_3, \dots, \omega_1 \leq \omega_2$, то это обозначается $\omega_1 \leq^* \omega_n$ и говорят, что ω_n сворачиваема к ω_1 .

Если заданная цепочка может быть сворачиваема к начальному символу грамматики, то она принадлежит языку, который порождается данной грамматикой.

6.2. КЛАССИФИКАЦИЯ ГРАММАТИК ПО ХОМСКОМУ

Американский математик Н.Хомский предложил классифицировать формальные языки по типу правил порождающей грамматики.

1) Грамматика класса 0.

Порождающие правила этой грамматики не имеют ограничений: $\phi \rightarrow \psi$, где $\phi, \psi \in (V_T \cup V_N)^*$.

Эта грамматика не имеет практического значения.

2) Грамматика класса I или *грамматика непосредственно составляющих* (НС – грамматика).

Это контекстно-зависимая грамматика, т.е. применение порождающих правил грамматики зависит от контекста:

$$\xi_1 \phi \xi_2 \rightarrow \xi_1 \psi \xi_2, \quad \xi_1, \xi_2, \phi, \psi \in (V_T \cup V_N)^*.$$

Пример контекстно – зависимой грамматики.

$$\begin{aligned} \Gamma_7: \quad V_T &= \{a, b, c, d\}, V_N = \{I, A, B\}, I, \\ R &= \{I \rightarrow aAI, AI \rightarrow AAI, AAA \rightarrow ABA, bBA \rightarrow bcdA, bI \rightarrow ba\}. \end{aligned}$$

3) Грамматика класса II или *контекстно-свободная грамматика* (КС – грамматика). Применение порождающих правил грамматики не зависит от контекста:

Порождающие правила этой грамматики имеют вид:
 $A \rightarrow \omega$, где $A \in V_N$, $\omega \in (V_T \cup V_N)^*$.

Пример контекстно – зависимой грамматики:

Γ_8 : $V_T = \{a, b\}$, $V_N = \{I\}$, I , $R = \{I \xrightarrow{1} aIa, I \xrightarrow{2} bIb, I \xrightarrow{3} aa, I \xrightarrow{4} bb\}$.
 $I \xrightarrow{1} aIa \xrightarrow{2} abIba \xrightarrow{1} abalaba \xrightarrow{4} ababbaba$ – зеркальный язык.

$L(\Gamma_8) = \{x\bar{x} : x \in V_T^*\}$, где $\bar{x}$ – зеркальное отображение x

4) Грамматика класса III или *автоматная грамматика*:

a) *левосторонняя автоматная грамматика*. Порождающие правила этой грамматики имеют вид: $A \rightarrow a$, $A \rightarrow Va$.

b) *правосторонняя автоматная грамматика*. Порождающие правила этой грамматики имеют вид: $A \rightarrow a$, $A \rightarrow aV$.
 $A, V \in V_N$, $a \in V_T$.

В одной автоматной грамматике не могут быть одновременно левосторонние и правосторонние правила.

Правила вида: $A \rightarrow aA$ или $A \rightarrow Aa$ называются рекурсивными.

Для каждой автоматной грамматики может быть построен реализующий её конечный автомат.

Пример автоматной грамматики:

Γ_9 : $V_T = \{a, b\}$, $V_N = \{I, A\}$, I , $R = \{I \rightarrow aI, I \rightarrow aA, A \rightarrow bA, A \rightarrow b\}$
 $L(\Gamma_9) = \{a^n b^m : a, b \in V_T, n > 0, m > 0\}$.

Язык, порождаемый грамматикой Γ_K , называется языком класса L_K .

Между формальными языками существуют следующие соотношения:
 $\{L_0\} \supseteq \{L_I\} \supseteq \{L_{II}\} \supseteq \{L_{III}\}$.

6.3. ПРИМЕРЫ ПОСТРОЕНИЯ ГРАММАТИК

Требуется построить грамматику, порождающую множество цепочек, состоящих из двух символов « $*$ », « $/$ » и каждая цепочка начинается с одной « $*$ » а заканчивается « $**$ », что является соответственно началом и концом цепочки. Между началом и концом может быть расположена произвольная последовательность « $/$ », разделенных « $*$ », т.е. допустимы следующие цепочки: $*/**$, $*/**/*$, $*/**/*/*$.

$V_T = \{/, *\}$, $V_N = \{I, A\}$, I .

Нетерминальный символ A будет определять любую непустую последовательность наклонных черточек « $/$ ».

$R = \{I \xrightarrow{1} * A **, A \xrightarrow{2} A/, A \xrightarrow{3} /\}$.

Многократное применение рекурсивного правила с номером 2 позволяет получить любую последовательность наклонных черточек.

$I \xrightarrow{1} * A ** \xrightarrow{2} * A/ ** \dots \xrightarrow{2} * A/// ** \xrightarrow{3} * A/// **$.

Для получения произвольного числа таких последовательностей в V_N добавляется нетерминальный символ B , который обозначает непустую

последовательность произвольного количества черточек, разделенных «*» и два правила: $V \rightarrow A$, $V \rightarrow V * A$.

Окончательно получается следующая грамматика:

$V_T = \{/, *\}$, $V_N = \{I, A, B\}$, I .

$R = \{I \xrightarrow{1} * B ** , B \xrightarrow{2} A, B \xrightarrow{3} B * A, A \xrightarrow{4} A / , A \xrightarrow{5} /\}$.

$I \xrightarrow{1} * B ** \xrightarrow{3} * B * A ** \xrightarrow{3} * B * A * A ** \xrightarrow{2} * A * A * A ** \xrightarrow{4} * A / * A * A ** \xrightarrow{4} * A // * A * A ** \xrightarrow{4} \dots \xrightarrow{4} * // * // * A // ** \xrightarrow{5} * // * // * // **$.

Примеры грамматик, используемых при определении языков программирования.

Терминальный словарь V_T – множество основных символов языка.

Нетерминальный словарь V_N – множество синтаксических понятий языка. Для описания синтаксиса языков программирования используется метаязык Бекуса или бэкусова – науровская форма (БНФ). В этой нотации вместо знака « $\rightarrow$ » (это есть) пишется знак « $::=$ ». Правила с совпадающими левыми частями объединяются в одно, а в качестве разделителя используется знак « $|$ » (или). Нетерминальные символы заключаются в угловые скобки « $<$ » и « $>$ ». В левой части БНФ пишется определяемое понятие, в правой цепочка символов и синтаксических понятий, либо только символы.

Пример определения в БНФ целого числа без знака:

$\langle \text{целое число без знака} \rangle ::= \langle \text{цифра} \rangle | \langle \text{целое число без знака} \rangle \langle \text{цифра} \rangle$
 $\langle \text{цифра} \rangle ::= 0 | 1 | 2 | \dots | 9$.

Пример грамматики для целого числа без знака:

$V_T = \{0, 1, 2, \dots, 9\}$, $V_N = \{I, A\}$, I ,
 $R = \{I \rightarrow IA, I \rightarrow A, A \rightarrow 0, A \rightarrow 1, \dots, A \rightarrow 9\}$.

Пример определения в БНФ идентификатора:

$\langle \text{идентификатор} \rangle ::= \langle \text{буква} \rangle | \langle \text{идентификатор} \rangle \langle \text{буква} \rangle |$
 $\langle \text{идентификатор} \rangle \langle \text{цифра} \rangle$
 $\langle \text{цифра} \rangle ::= 0 | 1 | 2 | \dots | 9$.
 $\langle \text{буква} \rangle ::= a | b | c \dots | z$.

Пример грамматики для идентификатора:

$V_T = \{0, 1, 2, \dots, 9, a, b, \dots, z\}$, $V_N = \{I, B, C\}$, I ,
 $R = \{I \rightarrow B, I \rightarrow IB, I \rightarrow IC, B \rightarrow a, \dots, B \rightarrow \dots z, C \rightarrow 0, \dots, C \rightarrow 9\}$.

Пример контекстно-свободной грамматики для идентификатора:

$\langle \text{идентификатор} \rangle ::= \langle \text{буква} \rangle \langle \text{конец идентификатора} \rangle$

$\langle \text{конец идентификатора} \rangle ::= \langle \text{буква} \rangle \langle \text{конец идентификатора} \rangle |$
 $\langle \text{цифра} \rangle \langle \text{конец идентификатора} \rangle | \langle \text{пусто} \rangle$
 $\langle \text{буква} \rangle ::= a | b | \dots | z$
 $\langle \text{цифра} \rangle ::= 0 | 1 | 2 | \dots | 9.$
 $V_T = \{0, 1, \dots, 9, a, b, c, \dots, z\}, V_N = \{I, B, C, E\}, I,$
 $R = \{I \rightarrow BE, E \rightarrow BE, E \rightarrow CE, E \rightarrow \Lambda, B \rightarrow a, \dots, B \rightarrow z, C \rightarrow 0, \dots, C \rightarrow 9\},$
 где Λ – пустой символ.

Пример построения грамматики, определяющей десятичное число со знаком и точкой.

$\langle \text{десятичное число со (зн.)} \rangle ::= + \langle \text{десят. Число со (зн.)} \rangle |$
 $\langle \text{десят. Число с (.)} \rangle | - \langle \text{десят. Число с (.)} \rangle | \langle \text{десят. Число с (.)} \rangle$
 $\langle \text{десят. Число с (.)} \rangle ::= \langle \text{целое число} \rangle . \langle \text{целое число} \rangle | . \langle \text{целое}$
 $\text{число} \rangle$
 $\langle \text{целое число} \rangle ::= \langle \text{цифра} \rangle | \langle \text{целое число} \rangle \langle \text{цифра} \rangle$
 $\langle \text{цифра} \rangle ::= 0 | 1 | \dots | 9$
 $V_T = \{0, 1, \dots, 9, +, -, .\}, V_N = \{P, Q, N, D\}, I,$
 $R = \{P \rightarrow + Q, P \rightarrow - Q, P \rightarrow Q, Q \rightarrow N . N, Q \rightarrow . N, N \rightarrow ND, N \rightarrow D,$
 $D \rightarrow 0, D \rightarrow 1, \dots, D \rightarrow 9\}$

Грамматика для выполнения арифметических операций.

Пример грамматики, определяющей множество правильно построенных арифметических выражений, использующих операции сложения и умножения над переменными обозначаемыми как « i ». Для задания порядка выполнения операций используются круглые скобки.

$V_T = \{i, +, *, (,)\}, V_N = \{E\}, E,$
 $R = \{E \rightarrow i + E, E \rightarrow i; E \rightarrow i * E, E \rightarrow (E)\}.$

Применение этих правил дает цепочку:

$E \Rightarrow i + E \Rightarrow i + E \Rightarrow i + (E) \Rightarrow i + (i * E) \Rightarrow i + (i * i)$

Набор этих правил не обеспечивает скобки для первых операндов, поэтому следует использовать следующие правила.

$R = \{E \rightarrow E + E, E \rightarrow E * E, E \rightarrow \epsilon, E \rightarrow i\}$

Применяя по-разному эти правила, получаем:

$E \Rightarrow E + E \Rightarrow E + E * E \Rightarrow \dots \Rightarrow i + i * i \quad [12444]$

$E \Rightarrow E * E \Rightarrow E * i \Rightarrow E + E * i \Rightarrow \dots \Rightarrow i + i * i \quad [24144]$

Для одной и той же цепочки получили два различных синтаксического разбора, следовательно, грамматика не однозначна. Другой недостаток операции сложения и умножения имеют одинаковый приоритет. Чтобы исключить равенство приоритетов операций можно использовать скобки:

$R = \{E \rightarrow (E + E), E \rightarrow (E * E), E \rightarrow i$

$E \Rightarrow (E + E) \Rightarrow (E + (E * E)) \Rightarrow (i + (i * i))$

$E \Rightarrow (E * E) \Rightarrow ((E + E) * E) \Rightarrow ((i + i) * i)$

Грамматика получилась неоднозначной, каждая операция оказалась в своей скобке и этим обеспечивается требуемая приоритетность операций.

Следующая грамматика учитывает приоритет:

$V_T = \{ i, +, *, (,) \}$, $V_N = \{ E, T, P \}$, E ,

E – арифметическое выражение, T – терм, P – множитель.

$R = \{ E \xrightarrow{1} E+T, E \xrightarrow{2} T, T \xrightarrow{3} T*P, T \xrightarrow{4} P, P \xrightarrow{5} (E), P \xrightarrow{6} i \}$

$E \xrightarrow{1} E+T \xrightarrow{2} T+T \xrightarrow{4} P+T \xrightarrow{5} (E)+T \xrightarrow{3} (E+T)+T*P \xrightarrow{2} (T+T)+T*P \xrightarrow{4} (P+P)+P*P \xrightarrow{6} \dots \xrightarrow{6} (i+i)+i*i$

Эта грамматика является однозначной и учитывает приоритет операций.

Рассмотрим обратную задачу – сворачивание цепочки: $i + i * i$ порожденной рассматриваемой грамматикой:

$i+i*i \xleftarrow{6} i+i*P \xleftarrow{6} i+P*P \xleftarrow{6} P+P*P \xleftarrow{4} P+T*P \xleftarrow{3} P+T \xleftarrow{4} T+T \xleftarrow{2} E+T \xleftarrow{1} E$.

Если в процессе сворачивания не учитывать старшинство операции умножения, то цепочка не свернется к начальному символу.

6.4. СООТВЕТСТВИЕ КОНЕЧНЫХ АВТОМАТОВ И АВТОМАТНЫХ ГРАММАТИК

Утверждение:

Для каждой автоматной грамматики может быть построен конечный автомат, допускающий язык, порождаемый заданной грамматикой.

Цепочка символов входного алфавита автомата называется *допустимой*, если под действием символов этой цепочки автомат из начального состояния переходит в одно из конечных состояний.

Множество допустимых цепочек автомата называется языком, допустимым данным автоматом.

Построение конечного автомата для заданной автоматной грамматики.

Преобразование автоматной грамматики:

1) множество нетерминальных символов V_N дополняется нетерминальным символом Z , который называется конечным;

2) правила вида: $A \rightarrow x$, где $A \in V_N$, $x \in V_T$, преобразуются к виду $A \rightarrow xZ$ для правосторонней грамматики или $A \rightarrow Zx$ для левосторонней грамматики;

3) к заданному множеству правил R добавляется правило: $Z \rightarrow \Lambda$, где Λ – пустой символ.

Построение графа автоматной грамматики и его разметка:

1) каждому нетерминальному символу ставится в соответствие вершина графа, которая помечается этим символом;

2) каждому правилу $A \rightarrow xB$, $A, B \in V_N$, $x \in V_T$ ставится в соответствие фрагмент графа, состоящий из дуги исходящей из вершины A , входящей в вершину B и помеченной символом x .

Соответствие понятий преобразованной автоматной грамматики и конечного автомата

1. Начальному символу грамматики соответствует начальное состояние автомата.
2. Алфавиту нетерминальных символов – алфавит состояний.
3. Алфавиту терминальных символов – алфавит входных символов.
4. Правилу $Z \rightarrow \Lambda$ соответствует только конечная вершина графа автомата.
5. Правилам грамматики соответствуют функции перехода автомата.

Пример построения автомата для следующей грамматики:

$V_T = \{c, d\}$, $V_N = \{I\}$, I ,

$R = \{I \rightarrow c, I \rightarrow Ic, I \rightarrow Id\}$.

$I \Rightarrow Ic \Rightarrow Idc \Rightarrow Id dc \Rightarrow cddc$

Цепочка растет справа налево, так как грамматика левосторонняя.

Грамматика после преобразования:

$V_T = \{c, d\}$, $V_N = \{I, Z\}$, I ,

$R = \{I \rightarrow Zc, I \rightarrow Ic, I \rightarrow Id, Z \rightarrow \Lambda\}$

Этой грамматике соответствует граф автомата рис. 6.2.

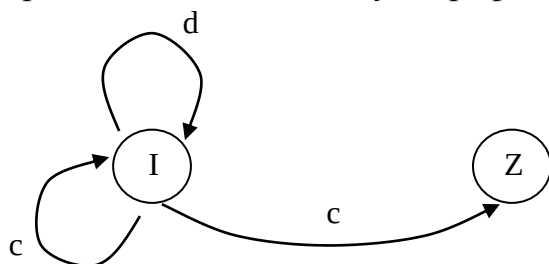


Рис. 6.2

Этот граф соответствует недетерминированному автомату, так как из состояния I под воздействием входного символа « c » существуют два перехода: из I в Z и из I в I . Подаем ранее полученную цепочку на вход автомата, начиная с крайнего правого символа, т.е. справа налево.

Для каждой левосторонней автоматной грамматики можно построить эквивалентную ей правостороннюю автоматную грамматику. Доказательство этого утверждения можно выполнить путем преобразования графа автомата, соответствующего левосторонней автоматной грамматике в граф автомата, соответствующего правосторонней автоматной грамматике и наоборот. Для этого необходимо поменять местами начальную и конечную вершину и изменить направление дуг на противоположное.

Проделав указанные преобразования с графом (рис.6.2), соответствующим левосторонней автоматной грамматике получим граф, соответствующий правосторонней автоматной грамматике (рис.6.3).

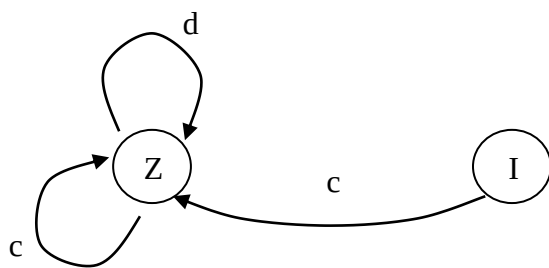


Рис. 6.3

Этому графу соответствует следующая правосторонняя автоматная грамматика:

$$V_T = \{c, d\}, V_N = \{I, Z\}, I,$$

$$R = \{I \rightarrow cZ, Z \rightarrow cZ, Z \rightarrow dZ, Z \rightarrow \Lambda\}$$

Построение из правосторонней автоматной грамматики левостороннюю автоматную грамматику выполняется аналогично.

Эти грамматики эквивалентны, так как они порождают один и тот же язык.

Построение грамматики по заданному автомату.

Пусть задан автомат без выходного преобразователя.

$$A = \langle P, S, s_0, \varphi \rangle$$

Для каждого конечного автомата можно построить автоматную грамматику, которая порождает язык, допускаемый заданным конечным автоматом. Для этого необходимо определить объекты грамматики следующим образом:

$V_T = P$ т.е. терминальному словарю ставится в соответствие входной алфавит;

$V_N = S$ словарю нетерминальных символов – множество состояний;

$$I = s_0.$$

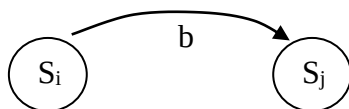


Рис. 6.4

Функции перехода на рис. 6.4 соответствует правило грамматики вида: $S_i \rightarrow bS_j$.

Пример построения грамматики по графу автомата рис. 6.5.

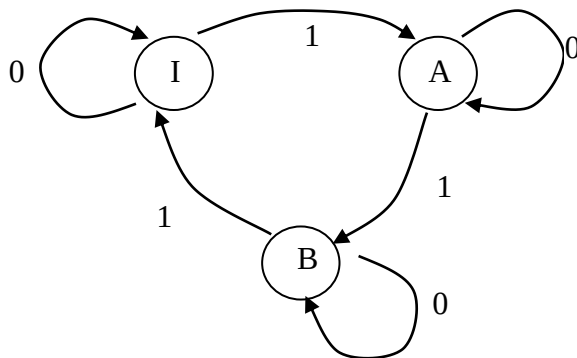


Рис. 6.5

Этому графу соответствует следующая автоматная грамматика:

$V_T = \{0,1\}$, $V_N = \{I, A, B\}$, I ,

$R = \{I \rightarrow 0I; I \rightarrow 1A, A \rightarrow 0A, A \rightarrow 1B; B \rightarrow 0B; B \rightarrow 1I\}$

В результате преобразования видно, что между автоматными грамматиками и автоматами существует взаимно-однозначное соответствие и следовательно грамматику можно рассматривать как одну из форм представления конечных автоматов.

6.5. НЕДЕТЕРМИНИРОВАННЫЕ КОНЕЧНЫЕ АВТОМАТЫ

Пусть A^0 – недетерминированный конечный автомат.

$A^0 = \langle P, S, s_0, \varphi^0, F \rangle$, где

P – входной алфавит;

S – множество состояний;

s_0 – начальное состояние;

F – множество конечных состояний;

φ^0 – функция перехода;

$\varphi^0: P \times S \rightarrow M(S)$, где $M(S)$ – множество всех подмножеств множества S .

Отличие недетерминированного автомата состоит в том, что функции перехода может определять не одно состояние, в которое переходит автомат, а некоторое подмножество состояний.

Если $L(A^0)$ – язык, который допускается недетерминированным конечным автоматом (НКА) A^0 , то существует детерминированный конечный автомат (ДКА) A , который допускает этот же самый язык, т.е. $L(A) = L(A^0)$.

Пример преобразования недетерминированного автомата в детерминированный.

Пусть задан недетерминированный автомат графом переходов (рис. 6.6).

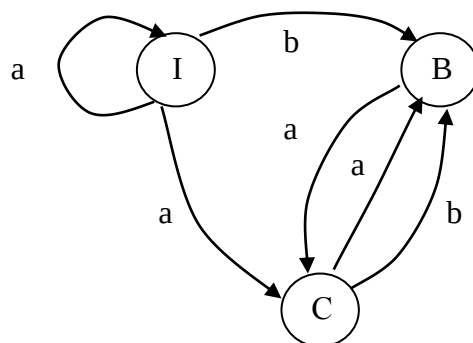


Рис. 6.6.

$P = \{a,b\}$, $S = \{I,B,C\}$,

$M(S) = \{[I], [B], [C], [IB], [IC], [BC], [IBC], \emptyset\}$.

Функции перехода автомата:

$\varphi(I, a) = [IC];$
 $\varphi(I, b) = [B];$
 $\varphi(B, a) = [C];$
 $\varphi(B, b) = [\emptyset];$
 $\varphi(C, a) = [B];$
 $\varphi(C, b) = [B];$
 $\varphi(IV, a) = [IC];$

$\varphi(IV, b) = [B]$
 $\varphi(IC, a) = [IC\ B]$
 $\varphi(IC, b) = [B]$
 $\varphi(BC, a) = [BC]$
 $\varphi(BC, b) = [B]$
 $\varphi(IBC, a) = [IBC]$
 $\varphi(IBC, b) = [B]$

Рассматривая каждое подмножество как новое состояние, строится граф детерминированного автомата (рис. 6.7).

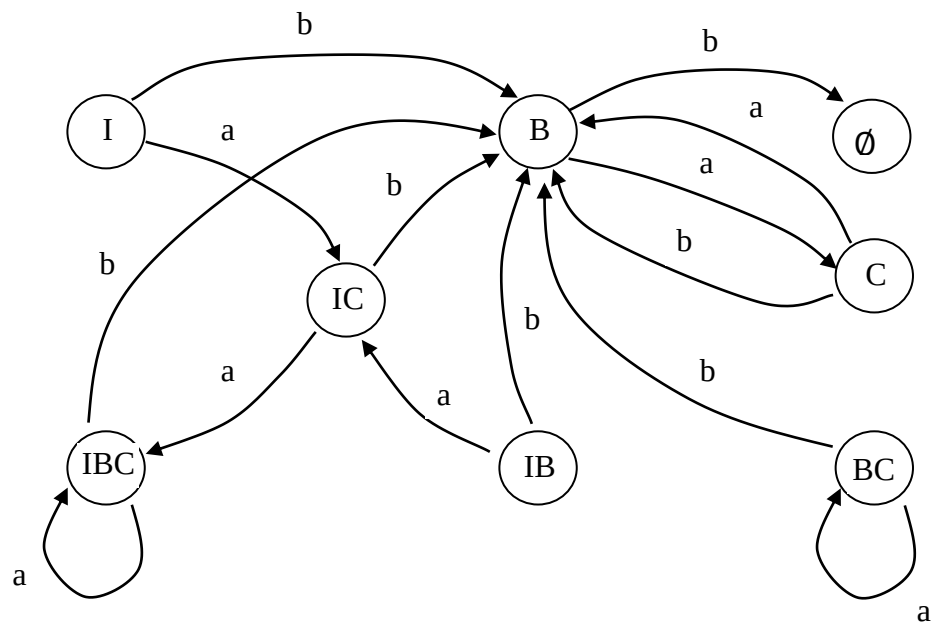


Рис. 6.7

Вершины IB и BC являются недостижимыми из начальной вершины I, поэтому они исключаются, упрощенный граф имеет вид (рис. 6.8):

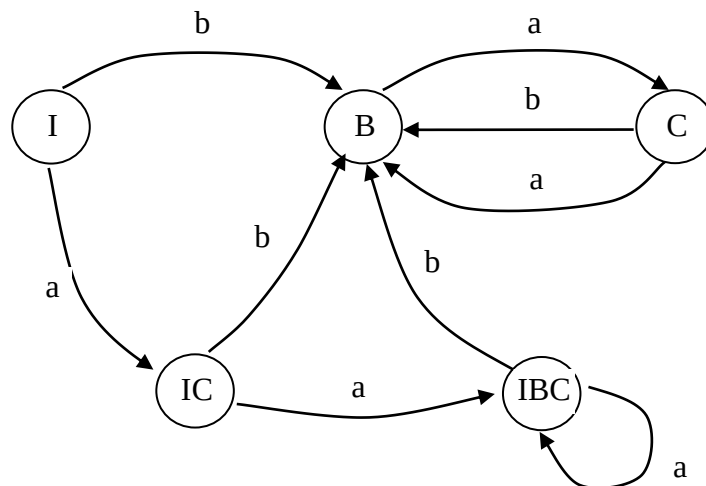


Рис. 6.8

Недостаток этого способа состоит в том, что в процессе преобразования возникает большое число недостижимых вершин, которые необходимо удалять.

Сокращенный способ перехода от недетерминированного автомата к детерминированному.

- а) строится заготовка таблицы переходов детерминированного конечного автомата;
- б) в качестве начального состояния (ДКА) выбирается начальное состояние (НКА);
- с) для каждого текущего состояния строится подмножество состояний, в которые переходит автомат;
- д) если при этом получаются состояния или их подмножества, которые отсутствуют в левой части таблицы переходов, то они включаются туда и осуществляется переход к пункту «с».

Построение таблицы переходов ДКА заканчивается, когда в левом столбце перестанут появляться новые подмножества состояний.

Пример перехода от НКА (рис.6.5) к ДКА по сокращенному способу с помощью табл. 6.1.

Таблица 6.1

$s_i \backslash p_i$	a	b
I	IC	B
IC	IBC	B
B	C	---
IBC	IBC	B
C	B	B

6.6. ПРЕОБРАЗОВАНИЕ НЕКОТОРЫХ ТИПОВ ГРАММАТИКИ К АВТОМАТНОМУ ВИДУ

Любой конечный язык, не содержащий пустой цепочки является автоматным языком, следовательно существует автоматная грамматика, порождающая данный язык.

Доказательство данного утверждения заключается в указании способа построения автоматной грамматики, порождающий данный язык.

Пусть задан язык:

$L = \{\tilde{x}_1, \tilde{x}_2, \tilde{x}_3, \dots, \tilde{x}_n\}$, где $\tilde{x}_i$ – цепочка и $\tilde{x}_i \in V_T^*$.

$\tilde{x}_i = a_1 a_2 a_3 \dots a_m$, где $a_i \in V_T$.

Для построения грамматики вводятся следующие нетерминальные символы: $A_1, A_2, \dots, A_{m-1}$ и для цепочки $\tilde{x}_i$ строятся следующие правила:

$\tilde{x}_i \rightarrow a_1 A_1, A_1 \rightarrow a_2 A_2, \dots, A_{m-2} \rightarrow a_{m-1} A_{m-1}, A_{m-1} \rightarrow a_m$

Применяя данную процедуру для каждой цепочки $\tilde{x}_i$, получим множество порождающих правил автоматной грамматики.

Контекстно-свободная (КС) грамматика называется *праволинейной* (*леволинейной*), если все её правила только праволинейны (*леволинейны*) или являются заключительными.

$A \rightarrow xB$ – праволинейное правило;

$A \rightarrow Bx$ – леволинейное правило;

$A \rightarrow \tilde{x}$ – заключительное правило, где $A, B \in V_N$, $\tilde{x} \in V_T^*$ цепочка символов терминального словаря.

Для любой праволинейной (леволинейной) грамматики может быть построено эквивалентная ей правосторонняя или левосторонняя автоматная грамматика.

Доказательство:

Пусть $\Gamma = \langle V_T, V_N, I, R \rangle$ праволинейная грамматика.

Так как грамматика праволинейна, то существует правило вида:

$A \rightarrow \tilde{x}B$, где $\tilde{x} \in V_T^*$.

Пусть цепочка: $\tilde{x} = a_1a_2\dots a_m$, где $a_i \in V_T$.

Тогда правило $A \rightarrow \tilde{x}B$ можно заменить правилом: $A \rightarrow a_1a_2\dots a_mB$. После ввода нетерминальных символов: $A_1, A_2, \dots, A_{m-1}$ данное правило можно заменить следующими правилами:

$A \rightarrow a_1A_1; A_1 \rightarrow a_2A_2; \dots A_{m-2} \rightarrow a_{m-1}A_{m-1}; A_{m-1} \rightarrow a_mB$.

Цепочка правил, заканчивающаяся $A_{m-1} \rightarrow a_mB$, заменит правило $A \rightarrow \tilde{x}$.

Применяя данную процедуру для каждого правила праволинейной грамматики, получаем правостороннюю автоматную грамматику.

Любую контекстно-свободную грамматику, содержащую правила вида: $A \rightarrow B$, $A, B \in V_N$ можно преобразовать в контекстно-свободную грамматику, не содержащую таких правил типа «нетерминал – нетерминал».

Доказательство: пусть задана контекстно-свободная грамматика, содержащую правила вида:

$R = \{\dots, A \rightarrow B, \dots, B \rightarrow Cx\}$.

В этом случае вывод любой цепочки, содержащий нетерминальный символ A , осуществляется следующим образом, если $\omega \Rightarrow \xi_1 A \xi_2$ и есть правило: $A \rightarrow B$, $B \rightarrow Cx$, то, применяя эти правила, в результате получим:

$\xi_1 A \xi_2 \Rightarrow \xi_1 B \xi_2 \Rightarrow \xi_1 Cx \xi_2$ это равносильно тому, что $A \rightarrow Cx$

Если в грамматике существуют правила:

$R = \{\dots, A \rightarrow B, \dots, B \rightarrow C, \dots, C \rightarrow z\}$.

В этом случае вывод: $\xi_1 A \xi_2 \Rightarrow \xi_1 B \xi_2 \Rightarrow \xi_1 C \xi_2 \Rightarrow \xi_1 z \xi_2$ и это равносильно тому, что $A \rightarrow z$

Алгоритм получения правил, не содержащих правил типа «нетерминал – нетерминал».

Пусть задана контекстно-свободная грамматика с множеством правил R .

1. Разобьем множество правил R на два подмножества R_1 и R_2 , причем в R_1 будут входить только правила типа: $A \rightarrow B$, где $A, B \in V_N$

2. Для любого символа A_i , стоящего в левой части правила, построим подмножество правил $S(A_i)$ следующим образом: если существует

$A_i \rightarrow A_n$; а из $A_n \rightarrow \eta B$, то в $S(A_i)$ включается правило: $A_i \rightarrow \eta B$.

3. Строится новая грамматика, содержащая следующий набор правил:

4. $R = \bigcup_{i=1}^k S(A_i) \vee R_2$ где k – число нетерминальных символов, находящихся слева в правилах подмножества R_1 . Построенная грамматика будет эквивалентно исходной и не будет содержать правил типа «нетерминал – нетерминал».

Пример преобразования грамматики:

$\Gamma: V_T = \{a, b\}, V_N = \{I, M, A, B\}, I,$

$R = \{I \rightarrow aM, M \rightarrow A, A \rightarrow aA, A \rightarrow B, B \rightarrow bB, B \rightarrow b\}.$

$R_1 = \{M \rightarrow A, A \rightarrow B\}.$

$R_2 = \{I \rightarrow aM, A \rightarrow aA, B \rightarrow bB, B \rightarrow b\}.$

$S(M) = \{M \rightarrow aA, M \rightarrow bB, M \rightarrow b\}.$

$S(A) = \{A \rightarrow bB, A \rightarrow b\}.$

$R^* = S(M) \cup S(A) \cup R_2 = \{M \rightarrow aA, M \rightarrow bB, M \rightarrow b, A \rightarrow bB, A \rightarrow b, I \rightarrow aM, A \rightarrow aA, B \rightarrow bB, B \rightarrow b\}.$

Контекстно-свободная грамматика может быть преобразовано к автоматному виду если она праволинейна или леволинейна и содержит правило «нетерминал-нетерминал».

Контекстно-свободная грамматика, содержащая одновременно праволинейные и леволинейные правила, не может быть преобразована к автоматному виду.

Контекстно-свободная грамматика с самовосстановлением, т.е. грамматика, которая содержит правило вида: $A \rightarrow \varphi A \psi$, где φ, ψ – непустые цепочки полного словаря ($\varphi, \psi \in V^*$ и $\varphi, \psi \neq \Lambda$), не может быть преобразована к автоматному виду.

6.7. ПОСТРОЕНИЕ РАСПОЗНАВАТЕЛЕЙ И ПРЕОБРАЗОВАТЕЛЕЙ, ВЫПОЛНЯЮЩИХ СИНТАКСИЧЕСКИЙ РАЗБОР

Распознаватель – это автомат без выходного преобразователя для определения, принадлежит ли заданная цепочка языку, порождаемому заданной грамматикой.

Построение распознавателя.

Пример построения автомата, распознающего во множестве трехбуквенных слов двоичного алфавита слова с двумя единицами: $L = \{011, 110, 101\}$, порождаемые следующей грамматикой:

$V_T = \{0, 1\}, V_N = \{I, A, B, C, D, E, Z\}, I,$

$R = \{I \rightarrow 0A, A \rightarrow 1B, B \rightarrow 1,$

$I \rightarrow 1C, C \rightarrow 1D, D \rightarrow 0,$

$$C \rightarrow 0E, E \rightarrow 1\}.$$

На рис. 6.9 представлен автомат, распознающий принадлежность слова языку, где входной алфавит $P = V_T$, множество состояний $S = V_N$.

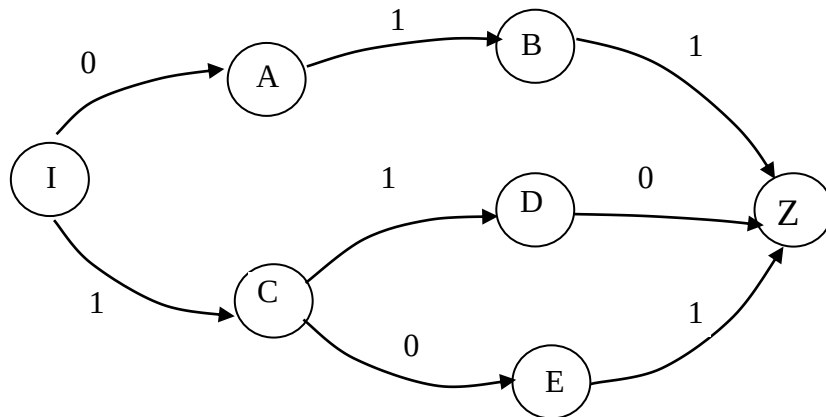


Рис. 6.9

Пример построения распознавателя допустимых и недопустимых слов для предыдущей задачи.

Для решения этой задачи необходимо множество всех слов разбить на подмножество допустимых слов L_1 и на подмножество недопустимых слов L_2 , а также ввести два заключительных состояния: Z_1 – для допустимых слов и Z_2 – для недопустимых слов.

Множество правил R необходимо дополнить правилами R' , порождающими недопустимые слова: $L_2 = \{000, 001, 010, 100, 111\}$.

$$R' = \{A \rightarrow 0F, F \rightarrow 0, \\ F \rightarrow 1, \\ A \rightarrow 1B, B \rightarrow 0, \\ E \rightarrow 0, \\ D \rightarrow 1\}.$$

$$V_N = \{I, A, B, C, D, E, F, Z_1, Z_2\}.$$

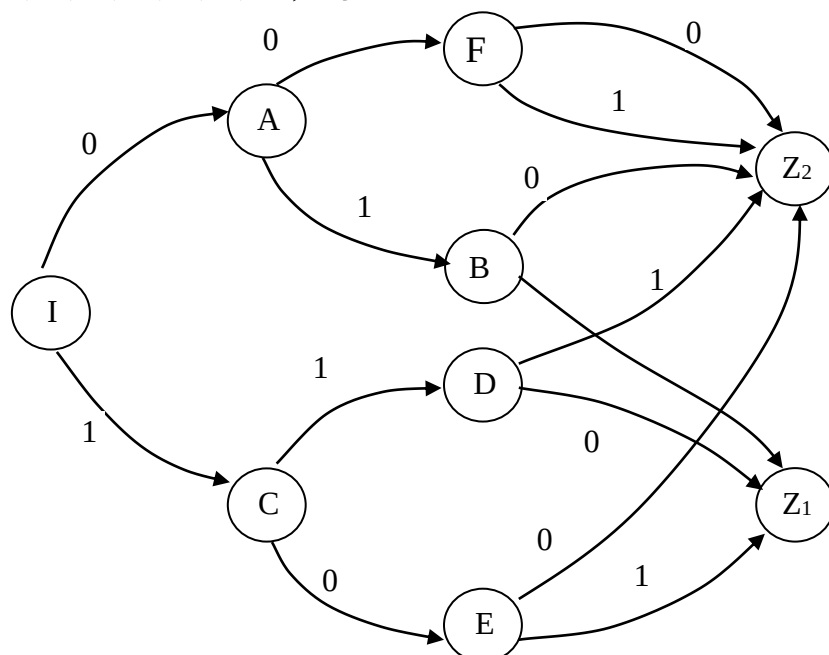


Рис. 6.10

Если в результате переработки трехбуквенного двоичного входного слова автомат (рис. 6.10) переходит в заключительное состояние Z_1 , то это слово допустимо. Переход в заключительное состояние Z_2 , показывает недопустимость входного слова.

Построение преобразователя, выполняющего синтаксический разбор.

Если задана грамматика, в которой все правила пронумерованы, то последовательность номеров этих правил, примененных для порождения заданной цепочки, называется синтаксическим разбором этой цепочки. Преобразователь – это автомат с выходным преобразователем, т.е. у него есть выходной алфавит и функция выхода, он осуществляет синтаксический разбор цепочек.

Пусть задана грамматика:

$$\Gamma = \langle V_T, V_N, I, R \rangle$$

Требуется построить автомат, выполняющий синтаксический разбор

$\Pi = \{P, S, s_0, W, \varphi, \psi\}$, где P, W – входной и выходной алфавит;

S – множество состояний, s_0 – начальное состояние;

φ, ψ – функции переходов и выходов автомата.

$$P = V_T, S = V_N, s_0 = I$$

φ, ψ определяются правилами грамматики R .

Последовательность действий для построения преобразователя.

По заданной грамматике строится распознающий автомат.

В качестве выходного алфавита выбирается множество номеров правил грамматики.

Строится множество функций выходов преобразователя, т.е. каждое правило с номером m вида:

$$A \xrightarrow{m} bC, \text{ где } A, C \in V_N, b \in V_T$$

будет определять в автомате функцию выхода, которой соответствует фрагмент графа на рис. 6.11.

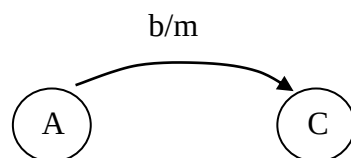


Рис.6.11

Пример:

$$V_T = \{a, b, c\}, V_N = \{I, A, B\}, I,$$

$$R = \{I \xrightarrow{1} aA, A \xrightarrow{2} aA, A \xrightarrow{3} bB, B \xrightarrow{4} bB, B \xrightarrow{5} c\}.$$

$$W = \{1, 2, 3, 4, 5\}$$

По приведенному выше алгоритму строится автомат, граф которого приведен на рис.6.12.

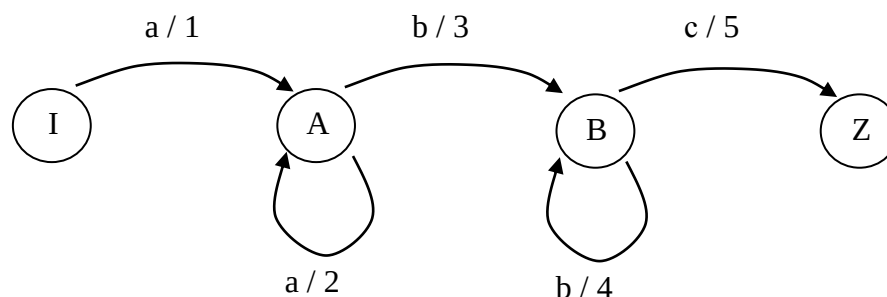


Рис.6.12

Применяя правила заданной грамматики, можно построить цепочку aabbbbc.

$I \stackrel{1}{\rightarrow} aA \stackrel{2}{\rightarrow} aaB \stackrel{3}{\rightarrow} aabB \stackrel{4}{\rightarrow} aabbB \stackrel{4}{\rightarrow} aabbbB \stackrel{5}{\rightarrow} aabbbbc.$

Если подать на вход автомата, граф которого приведен на рис.6.11, последовательность символов aabbbbc, то получим синтаксический разбор [1, 2, 3, 4, 4, 5].

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Глушков В.М. Синтез цифровых автоматов. – М.: Физматгиз, 1962.
2. Закревский А.Д., Поттосин Ю.В., Черемисинова Л.Д. Логические основы проектирования дискретных устройств. – М.: Физматлит, 2007.
3. Синтез конечных автоматов: учеб. пособие / В.С.Дудкин, Р.Ш.Исмаилов, А.В.Крайников; под ред. В.Б.Смолова. – Л.: ЛЭТИ, 1987.
4. Баранов С.И. Синтез микропрограммных автоматов (граф – схемы и автоматы): - 2-е изд., перераб. и доп. – Л.: Энергия, 1979.
5. Лебедев В.Н. Введение в системы программирования.- М., «Статистика»,1975

ОГЛАВЛЕНИЕ

1.	КОНЕЧНЫЕ АВТОМАТЫ	3
1.1.	Введение	3
1.2.	Теория конечных автоматов	3
1.3.	Автомат без выходного преобразователя	4
1.4.	Автомат с выходным преобразователем	7
1.5.	Эквивалентность автоматов Мили и Мура	13
2.	КАНОНИЧЕСКИЙ МЕТОД СИНТЕЗА КОНЕЧНЫХ АВТОМАТОВ	19
2.1.	Абстрактный синтез конечных автоматов	19
2.2.	Минимизация числа состояний полностью определенного автомата	27
2.3.	Минимизация числа состояний частичного автомата	31
3.	СТРУКТУРНЫЙ ЭТАП СИНТЕЗА КОНЕЧНОГО АВТОМАТА	38
3.1.	Типы элементов памяти	40
3.2.	Пример структурного синтеза синхронного автомата	43
3.3.	Кодирование состояний синхронного автомата	50
3.4.	Структура автомата с дешифратором	57
3.5.	Построение комбинационной схемы автомата на ПЗУ и ПЛИС	59
4.	СИНТЕЗ МИКРОПРОГРАММНОГО АВТОМАТА	63
4.1.	Построение графа автомата Мура по ГСА	63
4.2.	Построение не полностью определенного автомата	66
5.	АСИНХРОННЫЙ АВТОМАТ	70
5.1.	Синтез асинхронного автомата	72
5.2.	Состязание элементов памяти	73
5.3.	Кодирование состояний асинхронного автомата	75
5.4.	Синтез схем элементов памяти	80
5.5.	Исключение состязаний элементов памяти в синхронном автомате	98
5.6.	Построение комбинационной схемы асинхронного автомата	105
6.	АВТОМАТЫ, ФОРМАЛЬНЫЕ ЯЗЫКИ И ГРАММАТИКИ	118
6.1.	Основные определения	118
6.2.	Классификация грамматик по Хомскому	120
6.3.	Примеры построения грамматик	121
6.4.	Соответствие конечных автоматов и автоматных грамматик	124
6.5.	Недетерминированные конечные автоматы	127
6.6.	Преобразование некоторых типов грамматики к автоматному виду	129
6.7.	Построение распознавателей и преобразователей, выполняющих синтаксический разбор	131
	БИБЛИОГРАФИЧЕСКИЙ СПИСОК	134
	ОГЛАВЛЕНИЕ	135

Учебное издание

В.М. Пестриков, В.С. Дудкин, Г.А. Петров

ДИСКРЕТНАЯ МАТЕМАТИКА

Учебное пособие

Редактор и корректор Т.А. Смирнова
Техн. редактор Л.Я. Титова

Темплан 2013 г., поз. № 107

Подп к печати 18.11.13. Формат бумаги 60х84/16. Бумага тип. №1.
Печать офсетная. 8,75 печ.л., 8,75 уч.-изд.л. Тираж 200 экз. Изд. №107.
Цена «С». Заказ

Ризограф Санкт – Петербургского государственного технологического
университета растительных полимеров, 198095, СПб., ул. И. Черных, 4.